Uncertainty, Stress and Decision Simulation

Phase I Final Report
for the period of
5/1/00-9/30/00
(Unclassfied)

Under Contract: N00014-00-M-0069
Navy Small Business Innovation Research Program

Office of Naval Research

Prepared by:

Walter Warwick
Stacey McIlwaine
Patricia McDermott

Micro Analysis and Design
4900 Pearl East Circle Suite 201E
Boulder, CO 80301

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT DATE	3. DATES COVERED *(From - To)*
26-10-2000	Phase I Final Technical Report	01-05-2000 to 30-09-2000

4. TITLE AND SUBTITLE

Uncertainty, Stress and Decision Simulation

5a. CONTRACT NUMBER

N00014-00-M-0069

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Warwick, Walter
McIlwaine, Stacey
McDermott, Patricia

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Micro Analysis & Design
4900 Pearl East Circle, Suite 201E
Boulder, CO 80301

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
800 N. Quincy Street
Arlington, VA 22217-5660

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER

12. DISTRIBUTION AVAILABILITY STATEMENT

Approved for public release; SBIR report, distribution unlimited.

13. SUPPLEMENTARY NOTES

Prepared in cooperation with: Klein Associates, Inc.
1750 Commerce Center Blvd. North
Fairborn, OH 45324-3987

14. ABSTRACT

Please see attached.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Walter Warwick
U	U	U	UU	43	19b. TELEPONE NUMBER *(Include area code)* (303) 442-6947 ext. 146

Abstract

In May of 2000, Micro Analysis and Design, Inc. and Klein Associates, Inc., were awarded a Phase I SBIR to research and develop computational models of decision making in stressful and uncertain conditions (Topic # N00-074, Modeling and Simulation of Decision-making Under Uncertainty). This research was motivated by the need for improved behavioral realisms in computer generated forces (CGFs) with an eye toward reducing and, perhaps, even eliminating the need for human-in-the-loop simulations. Rather than continue in the tradition of rational choice theories and rule-based expert systems, we took a novel approach to this research and began work on a model of Recognition Primed Decision making (RPD). The RPD model explains how people can use their experience to arrive at good decisions without having to compare the strengths and weaknesses of alternative courses of action. For this reason, RPD theory seems to be a natural foundation for a more realistic model of human decision making under stress and uncertainty, but it also presents novel challenges from a computational point of view. We summarize below how we addressed these challenges during our Phase I work, and how we will work toward a validated modeling technology under a Phase II contract.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

Computer simulation is now commonplace in both training and systems analysis. While this development has mitigated the risk inherent in live exercises and prototype development, it has also revealed the shortcomings of the computational representations of human decision making behavior. These shortcomings are particularly clear in Computer Generated Forces (CGFs). Computer generated entities often behave predictably and inflexibly (Gillis and Hursh 1999) and this limits their value both as training tools and as valid predictors of human/system performance. The goal of our Phase I research was to research and develop modeling technologies to improve the computational representation of human decision making in stressful and uncertain conditions. Ultimately, we expect to improve behavioral realisms in CGFs and thereby reduce or even eliminate the need for human-in-the-loop simulations.

Our Phase I research has been premised on the assumption that the best way to improve behavior in CGF's is to bring the computational representation of human decision making more in line with the actual human decision making process. Hence, we have looked to the field of Naturalistic Decision Making for inspiration and, in particular, at the theory of Recognition Primed Decision making (RPD). As its name implies, RPD theory posits *recognition* as the central mechanism in the decision making process and thus stands in sharp contrast to more traditional theories of decision making that revolve around explicit, rule-based strategies. Although there is ample evidence that humans rarely use rule-based strategies in stressful and uncertain situations, most of the current representations of human decision making are rule-based. The resulting models are often brittle, costly to develop, and, most significantly, offer little insight into the human decision making process. Our work has been motivated by the need for a cost-effective model of decision making that is both robust and perspicuous.

We have leveraged our Phase I efforts off an independent (and ongoing) contract we have with NAWCTSD to develop a computational model of RPD (Developing Computational Models of Naturalistic Decision Making; contract # N61339-99-C-0103; COTR: Dr. Denise Lyons). That work was an important starting point both because it set the stage for us in general theoretical terms and because it provided the basic architecture we are currently using to model decision making in stressful and uncertain environments. But it was merely a starting point; the last six months of work has revealed a unique set of challenges. Below we will describe how we addressed these challenges during Phase I and how we foresee Phase II work proceeding. We begin in Section 2 with a discussion of our theoretical view of stress, uncertainty and decision making. Suffice it to say, the literature on this topic is somewhat fractured, and we have invested a good deal of effort to arrive at what we feel is a cogent theoretical foundation. In Section 3, we will describe some of the computational issues we have encountered. These are issues that go beyond the problems we faced when we first began working on our computational model of RPD. In section 4 we will discuss how these issues played against each other as we summarize seriatim the tasks we accomplished during our Phase I work. In Section 5, we will introduce preliminary results and describe some outstanding problems. Finally, in Section 6 we will summarize our work and offer a brief assessment of our progress

against the current state of the art. This discussion will set the stage for our proposed Phase II efforts.

2. Theoretical Issues

Stress and uncertainty are ubiquitous in the battlefield, but they are conspicuously absent from the behavioral representations in CGFs. It seems obvious that one way to improve behavioral realism in CGFs is to model the effects of stress and uncertainty. Indeed, this was the main thrust of our Phase I research. But despite the fact that the problem seems so obvious, the solution is very difficult. In order to model the effects of stress and uncertainty on the decision making process, we first had to understand these effects from a theoretical point of view. Unfortunately, we found a discouragingly disconnected theory; in fact, Hammond claims it is "internally incommensurable" (Hammond 2000). Nevertheless, we needed a starting point, and so we carefully picked through the literature. This section provides a brief introduction to the theoretical foundation we have adopted. First we offer a description of the theories upon which the computational model is grounded, and then we describe the psychological effects we have implemented.

Within the stress literature, studies can be placed into three categories depending on how stress is approached. For more detail on these distinctions, please see (Lazarus 1966; Levine and Scotch 1970; McGrath 1970; Cox 1975; Appley and Trumbell 1976). In the first approach, stress is treated as a dependent variable for study. Stress is described in terms of an individual's response to stress. The second approach treats stress as an independent variable and describes stress in terms of the stimulus characteristics of disturbing or noxious environments. The third approach views stress as the reflection of a lack of "fit" between the individual and the environment. In trying to reconcile these different opinions, we found ourselves attracted to the Klein's (1996) view that there is no such *thing* as stress; rather it is construct we impose on the world to help make better sense of the effects we see around us. In Klein's words stress is an intervening variable between particular kinds of stimulus-response relations. Although some will find this view iconoclastic, it affords us a principled foundation for treating workload as a quantifiable surrogate for stress (see Section 3.3 below).

For the purpose of this project, stressors are defined as acute stressors—sudden and temporary stressors such as personal threat, time pressure, noise, task overload, and distractions (Schmitt and Klein 1996). We have not addressed prolonged stressors such as life stress, sleep deprivation, and exposure to heat or cold.

As to the effects of stress, the traditional view of arousal (whose effect is analogous to stress) is the Yerkes and Dodson (1908) inverted-U theory. This theory postulates that both high and low levels of arousal result in reduced performance, while moderate arousal levels result in increased performance. Lacey (1967) later contradicted this theory by showing that physiological, cognitive, and behavioral responses to stress varied across different situations and did not all correspond to the Yerkes-Dodson curve. Again, after reviewing these results, we found ourselves confronting seemingly irreconcilable research. So, rather than try to implement contradictory performance reactions, we limited our attentions to the cognitive reactions to stress (see below).

Finally, we had to adopt the decision making model that would be subject to the effects of stress and uncertainty. The RPD model was chosen for this project because it describes how experienced people actually make decisions in naturalistic settings of time pressure, conflicting goals, and dynamic conditions. The RPD model explains how people can use their experience to arrive at good decisions without having to compare the strengths and weaknesses of alternative courses of action. The claim is that people use their experience to "size up" a situation, and recognize it as typical. Typicality amounts to the recognition of goals, cues, expectancies, and most important, a courses of action (see Figure 1). Where classical decision theories postulate a rational agent who carefully considers a host of alternatives against a background of perfect information, RPD theory describes an agent poised to act who depends on his expertise to assess the available information and identify the first workable alternative under less than optimal conditions.

For these reasons, RPD seemed a natural choice for a model of decision making in uncertain and stressful conditions. But as our work progressed, we found ourselves doubting that seemingly natural choice. In fact, as we continued our literature review a dilemma began to emerge. On the one hand, we wanted to begin with a theory that describes what people actually do under conditions of uncertainty and stress. While there is a large body of research on the effect of stress and uncertainty on analytic decision making processes (most notably, Tversky and Kahneman 1974), it seems that in stressful and uncertain situations *outside the laboratory* the decision maker does not have the time or luxury to compare options systematically. Indeed, research conducted in naturalistic settings reveals that experts rarely use analytic strategies and we saw no point in building a computational model to reflect a decision strategy rarely used in actual situations. On the other hand, however, the RPD model is presented as an explanation of how experienced decision makers can perform well *in spite* of stress. Researchers have observed experienced decision makers using the RPD strategy in a variety of stressful and uncertain domains. Remarkably, these decision makers were able to adapt to stressful conditions of noise, uncertainty, and time pressure to perform effectively time and again. This raised the question of whether the RPD model is sensitive to the effects of stress. It seemed we could either build a model of a decision making strategy that was sensitive to the effects of stress but patently unrealistic or we could follow the current research on what people actually do and end up with a model completely insensitive to the effects we wanted to capture.

Fortunately, the dilemma resolved itself. Discussions at Klein Associates revealed several aspects of the RPD model that are sensitive to stress. Some of the following aspects are reported in (Klein, Schmitt et al. 1996). Others are postulations based on ongoing observations in naturalistic settings and require further study.

- A decision maker's ability to imagine how a particular COA might unfold is impacted by time pressure. In terms of the RPD model this means that the agents process of *mental simulation* will either be severely limited or avoided altogether. Also, environmental stressors may impact the ability to "self-talk" (i.e., distractions could reduce concentration required to visualize or talk through a mental simulation).

- Stress, especially uncertainty, could change information seeking behaviors. Perhaps it will direct information seeking to specific salient cues. This focus on a particular cue could result in a loss of other information as attention is narrowed.
- The introduction of other tasks such as self-management tasks could result in excessive workload for the agent and this could lead to "sloppy" recognition of the situation. In other words, the decision maker might end up relying on fewer cues to assess the situation and thus overlook fine distinctions between situations. The result here could be an artificially crude choice among COAs.
- Reduced working memory capacity (because of noise and self monitoring) would result in a decrease in the amount of cues the decision maker could remember at one time. This could result in a speed/accuracy tradeoff.

Many of the impacts of stress on RPD are process effects (how a decision maker comes to a decision) as opposed to product effects (the outcome of the decision). This is due partly to the fact that the RPD is a satisficing model. Decision makers, especially under stress, do not try to come up with the best possible decision, but rather, a decision that works in the situation. This could make it difficult to compare the products of decision making. Using a satisficing strategy an agent might come to the same decision with or without the effects of stress and uncertainty. The differences would be in the process. A similar issue with a satisficing model is determining what constitutes a "good" decision. Is a good decision one that has a positive output (i.e., the track was correctly identified, the enemy was shot down)? If so, on what basis do you compare decisions to determine which is better? If two different decisions such as *conduct a frontal assault* and *perform a defensive maneuver* both result in the enemy attack being halted and minimal casualties, which is the better decision?

Validation of our computational model will be a delicate matter in light of these issues. But we have not let this fact delay our progress. Rather, we have focused our attention on narrowed attention and working memory limitations as the most tangible process effects from a computational point of view, and we have begun to explore alternative avenues for model validation at the process level. After some serious theoretical discussion, we are now confident that the RPD model is an appropriate foundation for our work. Moreover, we have been able to exploit mechanisms implicit in our computational model of RPD to realize these process effects in a natural way. This was an unexpected development, but it has given us greater confidence that our work is on the right track.

3. Computational Issues

We began our Phase I work with some of the basic computational components already in place. Under our contract with NAWCTSD we had begun development of a computational model of RPD using Micro Saint as the base implementing technology. We also had ready access to a variety of workload models as well as a sophisticated model of information processing. In fact, at the outset we faced something of an embarrassment of riches; there were so many models and ideas floating around that it was

hard to envision the architectural relations between our decision making model and the component models of stress and uncertainty. After spending some time in this mire, we found ourselves embracing a newfound sense of pragmatism; our attention turned away from the most sophisticated component models we could use to those that simply got the job done. We also found a more streamlined architecture emerging than the one we originally proposed. At the same time, however, there were still points in the model that called out for more sophisticated development. We describe the resulting RPD model below, starting with a review of the component models, continuing with a discussion of the links between the component models before concluding with a discussion of a performance server architecture we propose to use to embed our decision models in HLA compliant simulations.

3.1 The Computational Model of RPD

As we indicated above, we were already working with Klein Associates and NAWCTSD to develop a computational model RPD when our Phase I work began. At that point, we had constructed a task network model to reflect the flow of activities depicted in figure 1 and we had spent a good deal of time thinking about how to construct computational analogues for notions like "situation awareness," "typicality" and "anomaly detection." The most pressing problem at that time was to overcome a seemingly unavoidable theoretical contradiction, namely, how can one build a faithful, computational representation of an explicitly non-analytic decision strategy like RPD? The answer was, of course, to impose the appropriate levels of abstraction; although the underlying code would be necessarily rule-based (indeed, programming languages are the ultimate expression of rule-based behavior), we felt that at a conceptual level, the model was sufficiently far removed from the lower-level implementation details to engender non-rule based behavior.

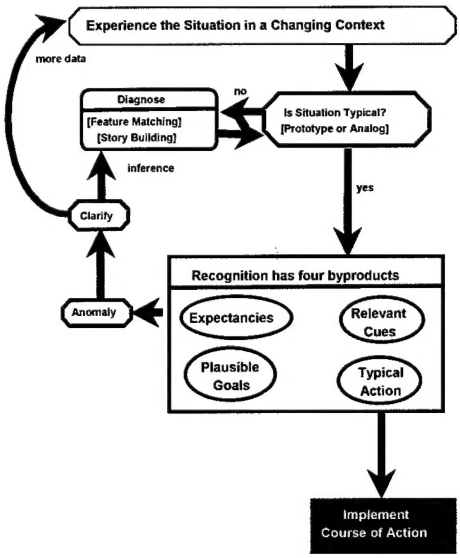

Figure 1. The RPD Model, Diagnostic Variation

From a theoretical point of view, this was an important step forward. But immediately after we had taken this step, we encountered another problem. Because we were so preoccupied with the implementation of the non-analytic aspects of the RPD, we hadn't paid very much attention to the fact that RPD is theory of *experienced* decision making. After working hard to address some difficult questions about situation awareness and knowledge representation, we happily settled for what turned out to be an overly simplistic approach to long term memory. In particular, we had decided to represent situations as sets of features, some of which represented low-level, brute facts about the environment, and others that represented high-level inferences and judgments the agent makes about his environment. We stored the encoded situations and their associated courses of action (COAs) as bit strings and stored them in a 2-D array with each row representing a particular episode. This so-called long term memory (LTM) array was to

be our computational analogue for experience. To make use of this experience, we developed a simple "recognition" routine in which we searched through the LTM array row-by-row to find the row that matched all the features of the current situation. Although it was straightforward, this approach to LTM had several shortcomings. First, LTM had to be populated by hand; before we could run a simulation, we would sit down and enumerate all the possible situations and decide a-priori which COA to associate with each situation. In this way, we guaranteed a match for every possible situation the synthetic decision maker might encounter. But it became clear during this process that some of the associations we were making were arbitrary— too many rows in LTM came to represent either meaningless or debatable associations between situations and COA's. Moreover, when we tried to disambiguate situations, we found ourselves engaged in a vicious regress. The seemingly obvious solution was to distinguish situations by introducing additional features that would, we thought, make for more fine grained distinctions. Of course, this meant more bits to encode, which meant more situations to distinguish and we found the very problem we had hoped to address recurring on a larger scale. This was bad enough, but in addition there was also something intuitively lacking in a recognition routine that simply matched features between a single row of LTM and the current situation. It seemed more plausible that recognition occurred as a host of similar experiences were recalled and not just when the features of a single episode matched those of the current situation. Finally, and most significantly, we began to worry that a recognition routine which depended on the hard-coded contents of LTM together with the recognition of a single episode would ultimately reflect, albeit indirectly, a form of rule-based behavior. By the time we began our Phase I work, it was clear that we needed a better model of LTM.

The better model came from an unlikely source. While reading up on situation assessment aids (Noble 1993), we came across a reference to (Hintzman 1986, Hintzman, 1984 #3) and his *multiple trace* model of LTM. Unlike other models of LTM that posit a store of generic concepts or *schema* (i.e. LTM as a collection of experience types), Hintzman suggests that LTM is simply a store of individual experiences (i.e., LTM as a collection of experience tokens). Each experience has its own trace in LTM, even if that experience happens to be exactly like another experience. The experiences themselves are represented by individual bit-strings that encode specific features of the experience. In these respects, Hintzman's work seemed similar to our own, but his recognition routine was significantly different. Indeed, rather than index a particular row of LTM as the product if recognition, Hintzman's model of recognition is a process of forming a composite "echo." The idea is to take something like a weighted average across *every* experience in LTM according to its *similarity* to the current situation. While it was clear that the basic structure of our original LTM matched Hintzman's multiple trace model, we hadn't considered anything like his recognition routine and we found the idea that recognition could be the product of multiple experiences quite compelling.

We were quick to adopt Hintzman's approach. We now talk about LTM as set of vectors (i.e., traces), each of which represents features of a situation, a set of expectancies about the situation and an associated COA. The expectancies and the COA are two of the four *by-products of recognition* (cf., Figure 1). Each vector element is three-valued: -1, 0, or 1.

Recognition depends on *similarity* values that are computed between the situation at hand and *each* trace in LTM. The similarity of a particular episode in LTM to the current situation is a normalized summation over the product of the situation features and the corresponding trace features. As Hintzman points out, the resulting sum is something like a Pearson's r-value where high similarity is reflected by values approaching 1 (although the analogy to a correlation coefficient isn't entirely complete since negative values, though possible, have no theoretical significance). Alternatively, in keeping with the vector terminology, similarity values can be thought of as dot-products that reflect the "alignment" of the current situation with a particular trace in LTM.

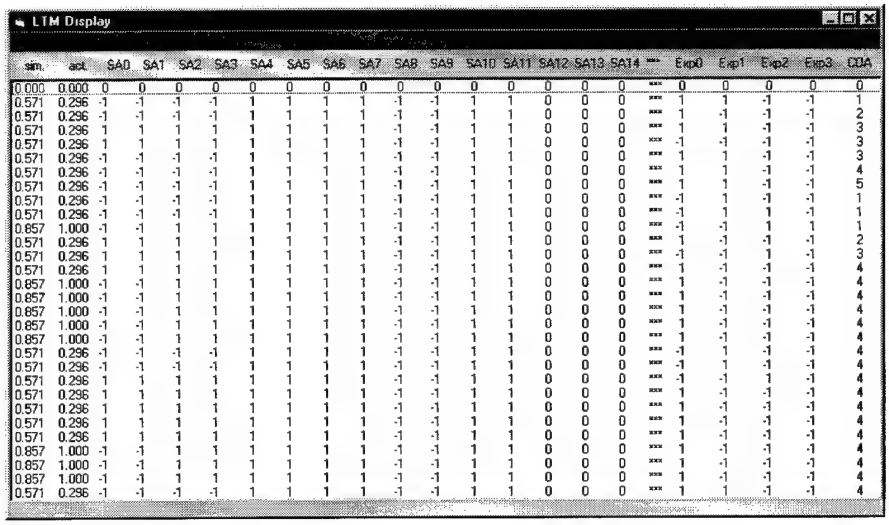

Figure 2 An example of LTM

These similarity values dictate[1] how much each trace in LTM contributes to the *echo* that represents the by-products of recognition. For example, suppose we are modeling a decision with two possible COAs: COA #1 and COA #2. To form the echo, we compare the current situation to all the traces in LTM and discover several vectors with high similarity to the current situation each of which has COA #1 associated with it. But we also find a few vectors that have COA #2 associated with them. The echo that comes back from LTM will strongly indicate COA #1 has been recognized, but there will also be hints of COA #2 in the echo. Figure 3 provides a visual example of an echo that is returned from LTM.

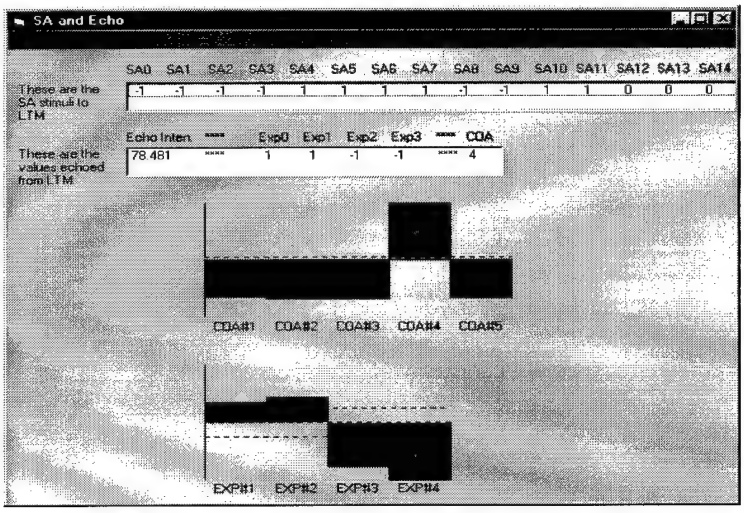

Figure 3 An example of an echo from LTM. In this case COA#4 is chosen; the other COAs are recognized, but they are all "remembered" as unsuccessful—hence the negative echoes.

[1] In the recognition algorithm, the contribution a trace actually makes to the echo is a cubic function of its similarity to the current situation. In this way we guarantee that a few highly similar traces will be better represented (i.e., more strongly recognized) than a larger number of less similar traces.

Because every trace in LTM can contribute to the echo, recognition is often "fuzzy." It will not always be clear which COA has been recognized since different traces might have different COAs associated with them. This is not too surprising given that Hintzman developed this multiple trace model to reflect aspects of associative recall—a phenomenon where a given stimulus will not always elicit a fixed response. Although we did not have associative recall in mind when we began to explore Hintzman's work, it turns out that having a "fuzzy" echo captures the intuition that was lacking in our original approach, namely, that when decision makers recognize a situation they are not matching it to a particular episode from their past but rather that they are recalling features from a host of experiences that were similar in important respects. Moreover, from the point of view of our current work, we believe that data structures and routines that underlie Hintzman's multiple trace model present us with a promising approach to models of uncertainty. For instance, given that the echo that comes from LTM might not be univocal, we have a natural analogue of ambiguity—a situation where the information about the current situation leads to different assessments of the situation. Conversely, if there is information missing about the current situation—perhaps the agent does not have access to a certain cue, or he doesn't trust his perception—we expect that the similarity values will be lower and less variable across the traces in LTM since there will be many traces with low similarity to the probe and no trace will be especially similar to the probe. Hence we expect that no particular COA will stand out in the echo. Again, we find intuitive appeal here; it makes sense that if an agent lacks information—for whatever reason—it will be difficult to pick out *any* course of action.

3.2 A Model of Uncertain Information (via LTM)

The results of a psychological study of uncertainty conducted by Klein Associates for the USMC (Klein, Schmitt et al. 1996) suggests that there are at least four sources of uncertainty that affect agents in three different ways. Among these, Klein Associates identified *missing information*, *unreliable information*, and *ambiguous information* as the kinds of uncertainty most frequently encountered during a combined arms exercise at both regimental and battalion command levels. These sources of uncertainty are consistent with the types of task-related stress identified in (Cannon-Bowers, Salas et al. 1996) as amenable to empirical study.

Although the multiple trace model of LTM afforded us a natural computational analogue for ambiguous and missing information, implementing these aspects of uncertainty required an analysis not present in Hintzman's original discussion. The main difficulty was that we could not tell just by inspection whether an echo represented the recognition of a particular situation or, rather, just the random convergence of otherwise unrelated traces in LTM. To put this worry into context, to model ambiguity with an echo containing multiple COAs, we had to know that the ambiguity was genuine in the sense that the various peaks in the echo were the product of multiple systematic associations in LTM and are not just random spikes in a noisy echo. Conversely, to model the effects of missing information we had to know when the information contained in the echo was lost in the noise. In short, we needed a way to filter the significant features of the echo from those that appear by chance.

Our approach to this problem was rather straightforward. We began by noting that, just like the traces in LTM, the echo from LTM is itself a vector composed of an ordered *n*-tuple representing the recognized course of action[2] followed by values representing whether the various expectancies for that situation have been asserted, denied or do not apply (values of 1, -1, and 0 respectively). To compute the individual values in the echo vector, we take a sum down the corresponding column of LTM, with each row in LTM making a contribution to the echo in proportion to its similarity to the current situation. Now, since any number of traces can contribute their (possibly different) COA to the echo, several bits in the COA portion of the echo might have non-zero values. This is the sense of a "noisy" echo; rather than find a single non-zero value in a particle bit-position, we might find several non-zero values in various bit-positions. Of course, if all the sufficiently similar traces agree on COA and expectancies, then we expect a clear, univocal echo. The more interesting question, however, is what we should expect if none of the sufficiently similar traces agree on COA or expectancies. Indeed, in the worst case, the association between situations and by-products would be entirely random, and the resulting echoes would be nothing but noise.

We have used this worst-case analysis to set a threshold between vector-values that can be ignored as noise and those that should be attributed to a systematic association between situations and by-products. In the case of the COA, a random association is naturally represented by a binomial distribution where the number of trials equals the number of traces in LTM and the likelihood of success for a given COA bit to contain a **1** is $1/m$ (m = the number of course of actions and therefore the number of slots in the *n*-tuple echo that are devoted to the COA). In the case of the expectancies, the likelihood of the expectancy being asserted *or* denied is .5 (i.e., it is equally likely that the bit contains something, a **1** or **–1**, as opposed to nothing, a **0**, in roughly half the traces). This gives us ready values for means and standard deviations and by fixing a confidence interval we can determine how many and how few 1's we expect to appear by chance in a given column. Given a mean value for the similarity across all the traces in LTM we finally arrive at threshold values that indicate the peak values we would expect if the echo were nothing but "noise."

These threshold values do three things. For the COA portion of the echo, they indicate when we have genuine ambiguity (as opposed to multiple non-zero values just due to chance). Second, they give us cutoff points for the expectancy values that reduce a continuously varying echo content value to a Boolean Value indicating whether the expectancy has been asserted (either positively or negatively). Third, by establishing a noise floor, we can determine when missing information about the current situation actually has an effect on recognition.

Finally, we should note the sense in which we claim to have modeled *uncertainty* here. In most discussions of decision making under uncertainty, the challenge is to develop algorithms that deal with missing or unreliable data. The approaches taken in those

[2] Where n is the total number of COAs available and the i^{th} COA is represented by a **1** in the i^{th} bit of the n-tuple and **0**'s in the remaining n-1 positions.

contexts often revolve around error and risk analysis, the propagation of subjective belief through a Bayesian network and the prospects for a clever search through an otherwise intractable solution space. In the real world, a decision making agent (human or computer) will always confront data that are both unreliable and capable of supporting multiple and even contradictory hypotheses. Developing computational systems to deal with real-world uncertainty is a daunting task. In the synthetic world of simulations, however, things are different. There are still data to contend with, but it is generated by us (or at least by our algorithms) and it is up to us to impose a sense of uncertainty. When does a synthetic tank driver doubt the information that appears, so to speak, on his synthetic range finder? When is a computer generated foot soldier unsure of which path to take through the computer generated terrain? The challenge here is to model the decision making behavior and the associated cognitive processes in such a way that the data available in the simulation can induce a sense of uncertainty in the limited "mind" of the synthetic agent. Before we can model the effects of uncertainty on decision making, we must know when the data are *perceived* as uncertain by the synthetic decision maker. The models we described above represent a first step toward endowing computer generated entities with these kinds of "subjective" responses and, in turn, more realistic behavior. Moreover, we believe our approach is both intuitively and theoretically satisfying. But it is important to keep in mind that knowing how to deal with uncertain data is very different from knowing when the data are uncertain.

3.3 A Model of Stress (via workload)

We did not have to look far for our model of uncertain information; there was an almost organic fit between the results of (Klein, Schmitt et al. 1996; Schmitt and Klein 1996) and the recognition mechanisms implicit in our model of LTM. Introducing the effects of stress, by contrast, was not so immediate. First of all, the theory concerning stress and its effects on the decision making process is messy. In fact, there is no univocal definition of stress in the literature, much less a uniform method for measuring it—whatever it is. Second, there wasn't an obvious analogue for stress in our model nor was there an obvious point at which to introduce its effects—whatever they are.

At the risk of stepping on theoretical toes, we began to survey workload measures as possible computational analogues for "stress." We did this for three reasons. First, given what little consensus there is about stress, workload stands out as a commonly cited example of endogenous stress (Hammond 2000). Second, we felt we could introduce workload requirements in a non-arbitrary way on the information-seeking tasks that drive decision making in our model. Finally, by equating stress and workload, we could tap into the established workload research methodologies that have proved useful to analysts in the past.

Among the several validated and accepted workload methodologies, the VACP (visual, auditory, cognitive and psychomotor) model developed by McCracken and Aldrich (1984) offered us the optimal combination of computational feasibility, theoretical validation, and algorithmic modesty. The VACP approach is simple and intuitive. Workload is a sum of parts—a multidimensional construct based on sensory, cognitive, and psychomotor components. The visual (V) and auditory (A) stimuli are attributes of the sensory component; the amount of information processing required of the operator

defines the cognitive component (C) and the agent's behavioral responses are what make up the psychomotor (P) component. McCracken and Aldrich developed 7-point ordinal scales for rating the visual, cognitive and psychomotor components and a 4-point scale for rating the auditory component of each task. These scales represent the relative difficulty for various component tasks. For example, on the cognitive scale an automatic task like simple association takes a value of 1.0 while more complicated, analytical tasks (evaluation, judgment, estimation, calculation) take on values between 6.0 and 7.0. The specific value for a task is based on the opinions of analysts and subject matter experts familiar with the task under consideration.

The VACP workload model indicates operator overloading in a straightforward way: Total workload for concurrent tasks is computed simply by summing the individual ratings of workload for all components (visual, cognitive, auditory, kinesthetic and psychomotor) involved in the task. If this sum exceeds a specified threshold, then the operator is overloaded. Of course, the actual value of this threshold and the operator's response to the overload are empirical matters and are relative to the other tasks involved. Nevertheless, the VACP methodology affords several advantages. First, because a single workload modality can induce an overload condition, workload estimates on the VACP model tend to be conservative. This provides us a hedge against inadvertently modeling the "super-operator," an agent capable of performing an unrealistic number of tasks simultaneously. Second, the VACP model has a solid empirical foundation (cf., Bierbaum, Szabo et al. 1987). Finally, implementation of the VACP model is reasonably straightforward. Although we have ready access to *WinCrew*, a computational model of Wickens' more sophisticated Multiple Resource Theory of workload, that code is very complex. Working on only a six month contract, we wanted to avoid the inevitable headache of integrating a stand-alone, commercial quality software product with our current test bed work.

3.4 Integrating the Component Models

After making ourselves familiar with the VACP methodology, we turned to the job of assigning the component V, A, C, and P measures to the information seeking tasks in our test bed model. The test bed model is centered around a driver's decision of whether or not to continue through or to stop at a stoplight. Information seeking tasks for which workload has been applied include checking the rear-view mirror, assessing the distance left until the stoplight, checking for cross traffic and so on. We relied solely on our own "expert" judgment, so the assignments of workload values were a bit *ad hoc*, but our concern at the time was not with empirical validity. Rather, with the workload measures in place we were finally in position to integrate the component models of stress, uncertainty and RPD.

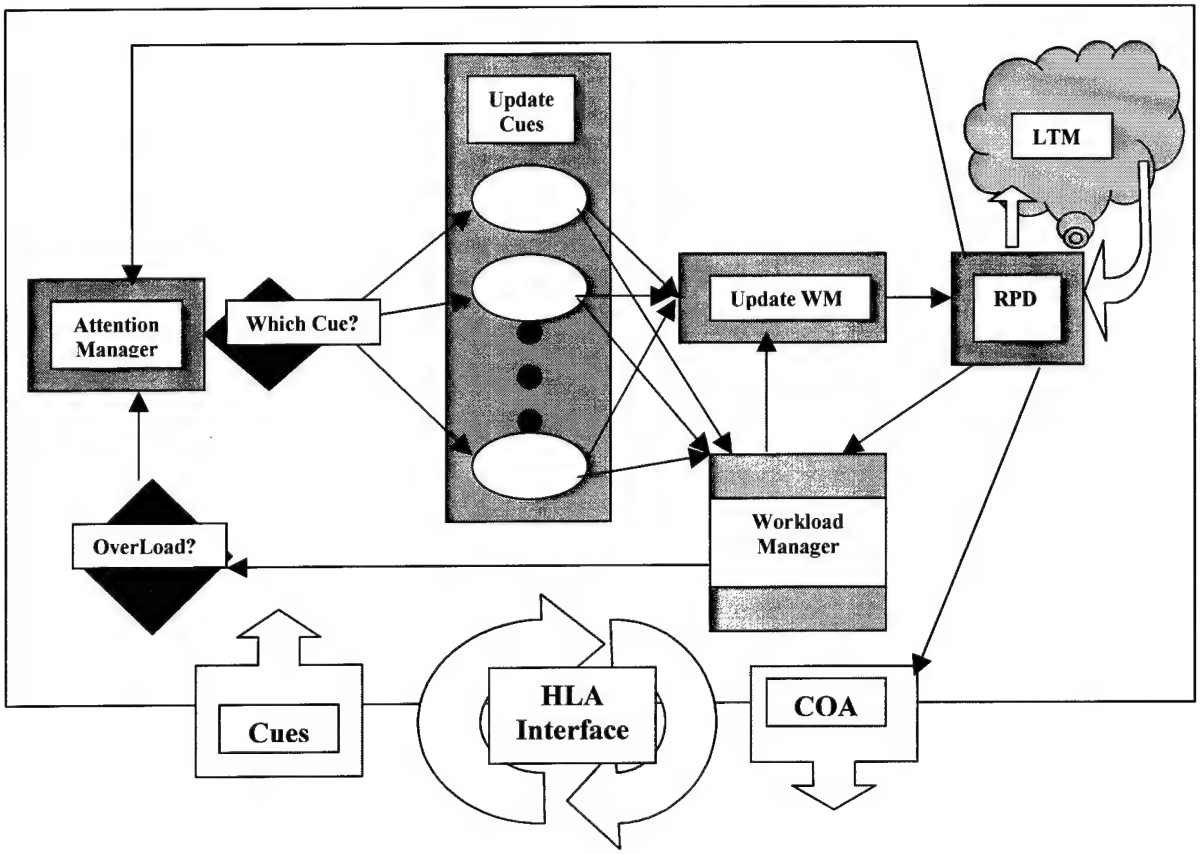

Figure 4 Architecture for the USADS Decision Making Model

The basic architectural relations are depicted in Figure 4 above. As discussed, the model of uncertain information is implicit in the echo returned from the LTM model to the RPD component. Likewise, the model of stress is embedded in the workload management component. The flow of activity across the model is as follows: A predetermined[3] set of cues from the "external world" is continually presented to the model across an (HLA) interface. The attention management routine imposes a sampling strategy according to the decision-maker's current goal and workload level. Sampled values are stored together with a time stamp in a "working memory" array. At the same time, the act of sampling contributes to an ongoing workload measure according to the VACP component value assigned to the associated cue update task. The updates continue (with values "decaying" toward 0 according to a default rate specified by the analyst) until a decision is required. At that point, the contents of the working memory array are used to probe LTM and an echo representing the by-products of recognition is returned. If it happens that the echo is uncertain or ambiguous (in the sense described above) then the cognitive workload component is increased, potentially affecting the sampling strategy used to update subsequent cues and the rate at which values decay in working memory is incremented to

[3] Determining which cues to present is a matter of balancing the information gained from a cognitive task analysis of the decision-type being modeled against the information actually available in the simulation.

the next faster decay rate all before the agent attempts to re-diagnose the situation (i.e., update his working memory and probe LTM again). Otherwise, the expectancies returned from LTM are used to direct subsequent cue sampling which continues until the expectancies are either satisfied, or the situation is re-diagnosed, or the decision maker simply runs out of time. In any event, the most recently echoed COA will be returned to the external world.

Throughout this process, we keep track of elapsed time. The agent is aware, so to speak, of the fleeting time to make his decision, and as time passes, the psychomotor workload component is continually incremented. This is intended to reflect a sort of "slow-boil" pressure on the agent and the attendant effects of self-monitoring that are likely to occur as time-pressure increases. As before, an overload condition induced by the psychomotor component will affect the sampling strategy used and the rate at which cues decay in working memory.

The effects of stress and uncertainty both play out in terms of narrowed attention and decreased capacity in working memory (insofar as increased decay rates effectively limit the amount of information that is likely to be present in working memory at a given time). In this way we preserve an important theoretical link between uncertainty and stress: namely, in addition to whatever effects uncertainty has on recognition, uncertainty itself is a stressor (cf., Klein, Schmitt et al. 1996)). Uncertainty induces both product and process effects. Where traditional approaches have emphasized the effects of uncertainty on product, our model of LTM allows us to capture both effects in an elegant manner: probing LTM with incomplete information will likely lead to imprecise recognition (i.e., noisy echo=bad product) while the echo itself can be analyzed in such a way to trigger increased workload components, and hence, contributes to increased stress and its attendant process effects. Indeed, there is an intuitively satisfying cycle represented here: uncertainty affects stress which, in turn, affects attention management and working memory, which can ultimately affect uncertainty.

Three comments are in order here. First, we have yet to implement the HLA-compliant interface (although we have begun to explore an HLA-compliant architecture; see below). During Phase I, we built a task network model of the external world around our decision making model, which made for a trivial exchange of data. We did, however, take care to encapsulate the decision making model so that it could be easily embedded in another simulation (task network or not). For example, array sizes and constant values can be read into the model from a separate text file. Second, how exactly attention shifts in a particular situation (i.e., which cues capture the agent's attention, which he will ignore) is to be determined by the analyst (see Section 4.6 below). We have yet to study the effects of shifting attention in our test bed model simply because we lack baseline data against which to compare decision output (this will be one of the first tasks we undertake should an option toward Phase II work be awarded). Finally, we have been conservative in the variety of effects we have implemented (though see Section 4.3 below). As we indicated above, the literature on the topic is difficult to sort out. There seems to be some agreement that attention does in fact narrow due to stress (Keinan 1987), and, by extension, that working memory capacity reduces (or is, at least, it is devoted to the

processing of fewer bits of information). Unfortunately, it remains an open question whether narrowed attention improves or degrades decision making performance (Klein 1996). It will be interesting to see how these process effects play out in our model. For the time being, however, we have steered clear of implementation just for the sake of implementation and have instead limited ourselves to modeling the process effects for which found some theoretical consensus.

3.5 An Interface to JSAF

CGF simulations have undergone extensive development and modification in order to implement new and advanced entity functionality. Like several other independent efforts that have aimed to improve entity representation, our Phase I work has focused on the possibility of enhancing a specific aspect in the behavior of a computer generated entity. As tantalizing as the possibilities might be, the improvement will go by the boards if we can't effectively integrate our models with a CGF like JSAF. The obvious solution is to incorporate our decision making models directly within the CGF. This, however, would only continue a trend of constant modification that has become something of a software maintenance dilemma for the JSAF integrators and would result in a highly complex and difficult to maintain product.

Fortunately, we can take advantage of another technique to add functionality in CGF models. We have begun to explore the process that involves transferring ownership of JSAF attributes to an external server. This technique expands the JSAF architecture to provide greatly enhanced entity representation without adding to an already complex CGF application. Under previous ModSAF development efforts, MA&D has developed a method that both dramatically improves the human/system performance representation and basically bypasses the configuration control process. That method of improving human representation is to transfer ownership of attributes, such as system/human performance, to an external server. More importantly, transferring ownership addresses complex modeling issues while minimizing or eliminating the impacts of configuration management on a software application such as JSAF.

We are currently exploring a performance server architecture that would allow us to realize the benefits of an improved model of decision making. Figure 5 shows the architecture. In the figure, the modules shown within the dotted box are part of the decision making server system. Each of the modules communicates and sends data using Distributed Interactive Simulation (DIS) protocol data units (PDUs) or High Level Architecture (HLA) objects, attributes, and interactions. The other modules are typical of those that would interact with the decision server and also communicate though a DIS/HLA network.

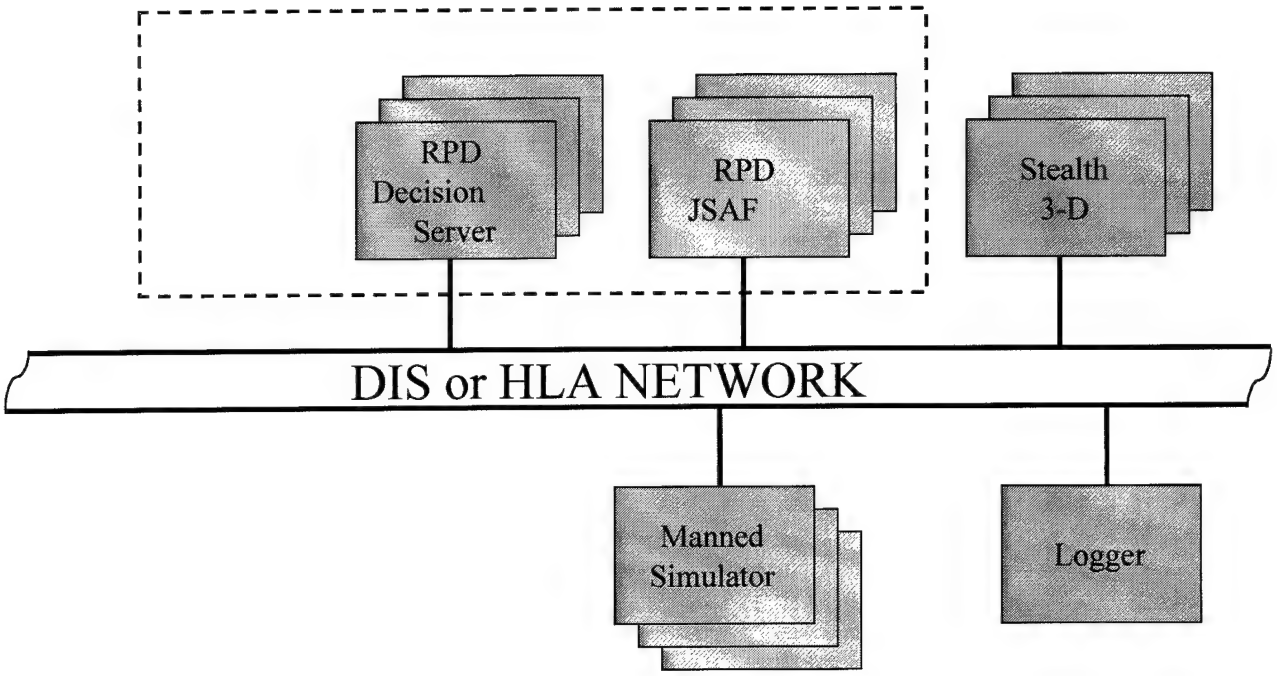

Figure 5 The RPD performance server architecture

The RPD Decision Server determines JSAF entity behaviors based on the algorithms and data structures discussed above. These decisions are then "served" to the JSAF entities. The other boxed module is the RPD-enabled JSAF. It is a slightly modified version of JSAF that can communicate and transfer attribute ownership to an external server (the decision server in this case). The RPD-enabled version of JSAF will retain all the performance capabilities found in the baseline version but will also have the ability to obtain realistic human decision making behaviors from the RPD Decision Server. This capability will be selectable by the JSAF operator through a graphical user interface (GUI). The following subsections provide specific details on each of these simulators.

3.5.1 The RPD Decision Server

The RPD Decision Server will track subscribed JSAF entities and operational performance to represent more realistically both the decision making process and product on the synthetic battlefield. The act of an RPD-enabled JSAF entity subscribing to a performance server transfers executions of the appropriate decision type from RPD-enabled JSAF to the RPD Decision Server. This subscription process causes the RPD Decision Server to begin tracking the subscribed entity. It records entity tracking information including its type and any provided parameters the analyst deems germane to the decision-making process. The RPD Decision Server then provides decision parameters to the subscribed entity.

3.5.2 RPD-enabled JSAF

The performance server architecture depends on an RPD-enabled version of JSAF. This means we must modify JSAF to represent decision making effects on entity performance and behaviors. In RPD-enabled JSAF, we will affect the behavior of a particular kind of

entity making a particular kind of decision. We will not attempt to modify every aspect of an entity's decision making behavior. As the chosen entities reach the selected decision points, RPD-enabled JSAF will affect these decision behaviors so that they represent actual human behaviors more accurately.

3.5.3 Connecting the Client to the Server

A subscription process will be used by the performance server architecture to perform the transfer of attribute ownership from an RPD-enabled JSAF entity to an RPD Decision Server. When RPD-enabled JSAF is used to create an entity, a process to subscribe to a RPD Decision Server will be initiated. The sequence will involve a handshake protocol that ensures that a link is established between an individual entity and a single RPD Decision Server. The subscription process is depicted in Figure 6.

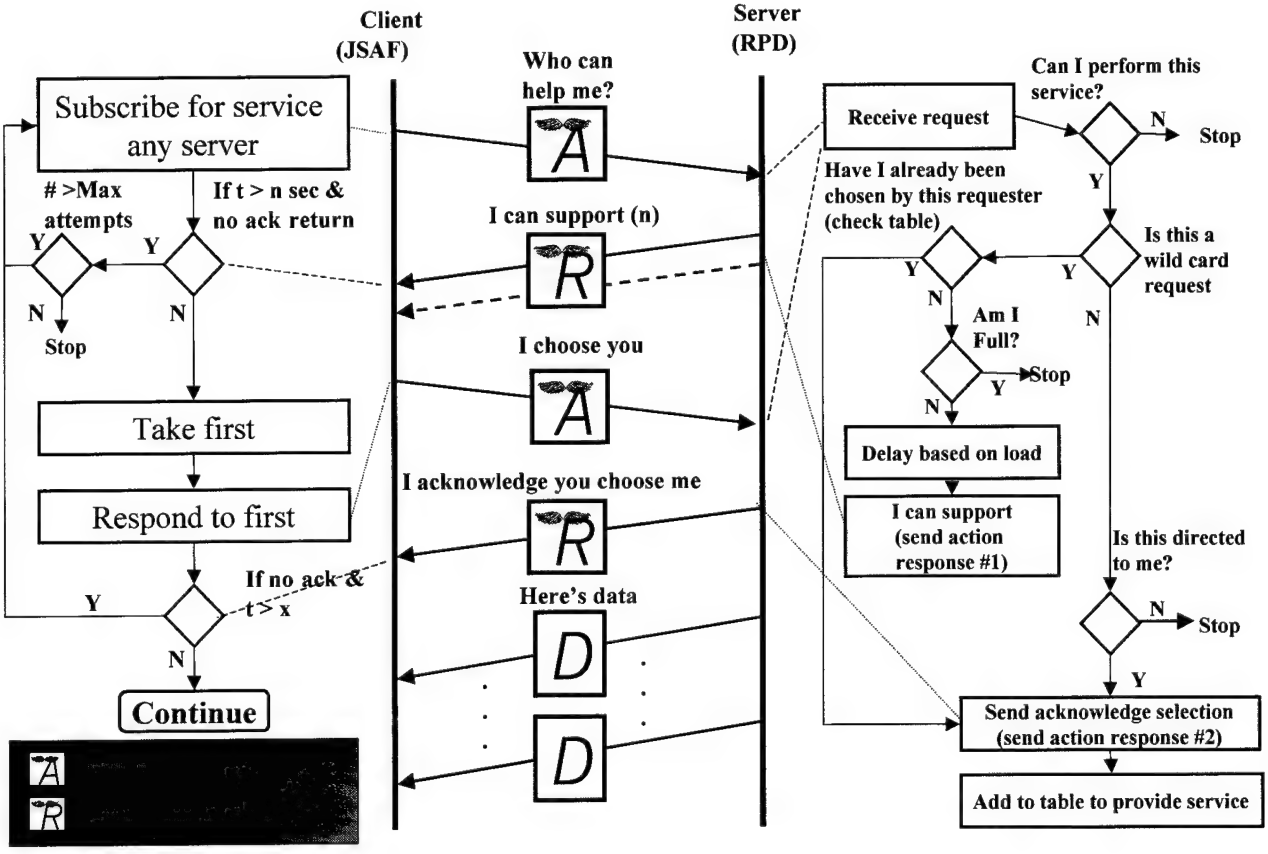

Figure 6 The subscription process for transferring attribute ownership from a client to a server

An RPD-enabled JSAF entity will initiate the subscription process with RPD-enabled JSAF sending an Action Request PDU requesting service from an RPD Decision Server. If available, an RPD Decision Server will reply with an Action Response PDU acknowledging that it can serve performance data to that entity. Because the architecture is scaleable, multiple RPD Decision Servers could provide such data, hence multiple RPD Decision Servers might respond to the request. RPD-enabled JSAF will arbitrate the

responses and select a specific RPD Decision Server by replying with an Action Request PDU. Along with the reply, the RPD-enabled JSAF entity will send any initialization data indicating prior stressor conditions the entity has encountered. The selected RPD Decision Server will acknowledge the reply and then will provide behavioral and operational performance parameters to the RPD-enabled JSAF entity.

This subscription process will effectively transfer ownership of the RPD-enabled JSAF entity's behavioral and operational parameters to an RPD Decision Server. As we indicated above, our architecture will be scalable and allow for any number of RPD Decision Servers to be present.

3.5.4 Embedded versus performance server approach

We have benefited from the experience of our previous efforts to improve entity behavioral representation in ModSAF. The performance server architecture we described above is one of the fruits of that experience. We propose to use this architecture rather than embedding our decision making models directly into JSAF for several reasons. First, implementing something as complex as decision making behavior will require extensive modifications to existing JSAF libraries and the inclusion of additional libraries and data files. Consequently, the embedded model would not provide much flexibility or ease of expandability. If we were to model additional decision types or if the existing ones required changes, we would have to develop additional software for the JSAF code. The embedded approach is also limited to a relatively simplistic implementation of the behavioral representation methodology. A more sophisticated approach could have the potential to significantly degrade JSAF performance. By contrast, the performance server architecture will allow us to have dynamic environments affect the behavioral representation of JSAF entities. Based upon previous ModSAF development experience, the performance server architecture will require fewer, easier, and simpler modifications to JSAF and performance will remain basically unaffected.

In addition, there will be benefits that do not relate to JSAF functionality. These benefits are wide ranging and have more of an effect on implementation and development within JSAF. They include improved configuration management of JSAF, simplified verification, validation and accreditation (VV&A), scalability, and improved behavior representation fidelity. Using the performance server approach, JSAF configuration management issues will be vastly reduced. Modification or additions to decision making behavior functionality will not involve JSAF and thus will not interact with the JSAF configuration management process. The external server will provide the gains in JSAF functionality and JSAF will only need to have the capability to use this server. VV&A issues will be reduced due to the fact that the functionality provided by the server will go through the VV&A process without directly involving JSAF. Currently, when functionality is added into JSAF, a VV&A process involves all of JSAF. Given the large and complex nature of JSAF, attempts to verify, validate and accredit enhancements to JSAF can be challenging. The performance server approach is scalable in that multiple servers can be used to service the demand. The ability for an entity to select a performance server that is not busy will be built into the subscription process. Part of this selection process will include load balancing algorithms to distribute entities and the processing load among servers. Finally, user selectable fidelity will also be a huge

benefit. In the embedded approach, all of the entities must use the same behavioral representation methodology. Under the performance server approach, the server behavioral representation algorithms can be modified much more easily and customized as desired. Individual servers can have different levels of fidelity and CGF users will have an improved capability to compartmentalize simulations into classified and unclassified cells.

While it has many benefits, the performance server architecture has two potential shortcomings. The first is the potential to increase network latency. If the behavioral representation or additional functionality being added via a performance server approach is extremely time sensitive, network latency could be unacceptable. Given our performance server application, we believe this latency will be small compared to the behavioral times and will thus have a negligible effect. The second potential shortcoming is increased network bandwidth. Due to the network communication between the server and JSAF, the network bandwidth needed will be increased. In our scenarios, given the length of time between requests for parameter updates from the server, we believe the additional bandwidth needed will be negligible.

4. Phase I Tasks

The technical approach we described above developed as we pursued the following Phase I tasks.

- Select real-world examples of decision making in stressful and uncertain environments
- Determine the primary modeling objectives
- Design an integrating architecture
- Identify gaps in modeling architecture
- Develop a functional specification for the model
- Develop a user-interface prototype for our modeling tool

With the exception of the fourth task, the completion of these tasks required both theoretical and computational effort. We describe below the extent to which we have accomplished each task and, moreover, how otherwise disparate research efforts in psychology and computer simulation have complemented each other throughout our Phase I work.

4.1 Task 1 – Select real-world examples of decision making in stressful and uncertain environments

Although the long-term goal of this project calls for the development of a generic technology to model a variety of decisions, our work began with a search for specific examples of decisions made in stressful and uncertain conditions. We did this for two reasons. First, from a theoretical standpoint, such examples give concrete meaning to terms like "typicality," "situation awareness," and "expectancy." By understanding how these terms apply in a specific setting, we gain a deeper understanding of the extent to which they can be generalized in a computational model. It is one thing to say that an agent will re-assess a situation in the face of a failed expectancy, but it is quite another to say what that really means in a specific operational context (does the agent re-consider the information he already has, does he look for new information? what, exactly, is the information under consideration in the first place? etc). The answer to such questions in specific settings gives us more detail to work with than theory alone provides. Second,

from the point of view of model development, concrete examples provide both an empirical touch stone to guide our efforts and offer one of the few potential avenues of validation for our decision models. Indeed, unlike other aspects of human performance, it is very hard to quantify the products of decision making behavior and thus we find ourselves relying on existing analyses of actual decision making behavior for both model construction and validation.

With these motivations in mind, we worked with Klein Associates to identify existing work from which we might draw our examples. In recent years, Klein Associates has conducted three major efforts with different branches of the United States Marine Corp (USMC). The first was to understand the decision requirements of Marine Corps in regimental command and control centers. Klein Associates was asked to study the structure of decision making in the Combat Operations Center (COC), in order to determine how to improve it. Four exercises were observed and over 200 critical decision-making incidents were collected from key players in the COC. The key decision requirements of the COC were identified as an organizational entity, as well as some of the primary barriers to carrying out these requirements. For example, the seasoned officers in the COC could size up situations rapidly, yet most procedures put the task of situation assessment in the hands of junior officers, who struggled because of their lack of experience. Another problem was in handling uncertainty. Frequently, actions were taken more to reduce anxieties about what was happening, rather than to answer real and timely questions. This project was a front-end analysis of the COC decision making, and the findings are being used in several different ways.

The second effort dealt with the USMC's experimentation with a new method of fighting battles that is designed to be safer, seamless, adaptable, and hard to counter. The concept consists of dispersing multiple squad-sized units on a battlefield and having these units be the "eyes-on", calling in reports of enemy activity and directing fires on the enemy from aircraft and from ground units in the rear. The theory here is that these small dispersed units are rather difficult for a mechanized enemy to destroy, let alone locate, and they can be inserted and removed quickly. Commanding and controlling these units is no small task. What type of control structure will work best? How do you portray this clearly on displays? How can technology support as opposed to hinder this type of operation? The USMC is working on such questions, and Klein Associates' role was to study the decision making that occurred (or did not occur) and find ways to support it. Klein Associates also provided an unbiased perspective on what supports effective operations and what does not in terms of displays, organization, ergonomics and information flow.

The third effort is related to the second effort. Given the re-organization described above, the squad leaders out on the battlefield are in new and vastly different roles than they have previously encountered. They are now being faced with challenging decisions that have never before been placed upon them. Klein Associates was tasked to train these enlisted Marines to become better thinkers and decision makers on the battlefield. The Decision Skills Training program was developed to accomplish this. The approach was to improve decision-making performance by providing tools that facilitate the development of decision makers' domain expertise, rather than teaching generic decision-making

strategies. Tools were developed to teach squad leaders to read situations better and faster, identify plausible options more effectively, manage their time and attention better, see patterns, make discriminations, and show more of the characteristics of expert decision makers. The tools were refined based on the feedback Klein Associates received from an experiment. Several methods were developed to provide increases in experience, and increases in the amount of learning that can be derived from experiences. These include a technique for identifying decision requirements, a method for presenting low-level simulations, a method for reflecting on the decision making in training events, a method for mentally simulating plans, and a method for leaders to obtain feedback on expression of intent.

Although none of these efforts were undertaken with a computational model of decision making in mind, the work is still relevant. In order to make recommendations about organization, training and decision support it was necessary to identify the stresses and types of uncertainty that the subjects faced in their various decision-making domains. It was likewise necessary to understand how agents deal with these demands. Both kinds of information are important to our current work.

Originally, we had intended to sift through this body of work in order to identify the kinds of decisions that would be most germane to ONR interests and, of those, select examples for which we have ready access to supporting data. We have, however, procrastinated our original plans. As we began to consider how we would embed our decision making models into JSAF, it became clear that the variety of "decisions" we might actually be able to affect will be somewhat limited. So, we thought it best to wait until we have conducted a more thorough survey of the JSAF code before we settle on specific decisions to model. Indeed, it makes no sense to model a decision that cannot be affected, or worse, does not exist, in the existing JSAF code. Once we have a better idea of the available decisions, we will select from them according to the same criteria discussed above.

In the meantime, we have piggybacked our development efforts on the test bed model we constructed for our work with NAWCTSD. The test bed model is built around a driver's decision whether to run a traffic light. We used this example because it struck us as naturalistic and because driving is a familiar domain (no need for SMEs). The model itself is fairly modest in its complexity, but work on the test bed has still demanded that we make theoretical issues concrete. Moreover, because the test bed is fairly simple, the implementation of the additional stress and uncertainty models has been relatively straightforward.

4.2 Task 2 – Determine the primary modeling objectives

It almost goes without saying that the design of a modeling technology depends on its intended use. For example, a tool designed to be embedded as an intelligent expert in an embedded training system would be quite different than a tool used to improve behavioral realism in CGFs. After preliminary discussions with ONR personnel, it became clear that a realistic model of decision making is needed to eliminate human-in-the-loop CGF simulations so that multiple runs can be performed quickly and cheaply. Moreover, the simulated decision behavior needs to be sufficiently realistic so that the

results from the simulation can be used to answer questions that resist more direct, analytical solutions.

Several design implications follow from this objective. First, the need for a validated predictive model takes on added importance in this context. In the training environment, the current lack of behavioral realism in CGFs has obvious, but limited, consequences. Most soldiers can quickly identify their computer generated opponents and while this limits the effectiveness of simulator-based training, it does not affect the other types of training a soldier will receive nor, for that matter, does it imply that simulator-based train is without merit. By contrast, the data produced by a simulation depend entirely on the robustness of the underlying models. If those models are suspect, then so too is the data gained from them. Unlike the use of CGFs for training, the use of CGFs for analysis is an all-or-nothing game. Second, a tool that supports the use of CGFs for analysis should not significantly impact the time or effort it takes to develop the overarching CGF scenarios. The tool should allow for the efficient construction and integration of decision making models, and the resulting models themselves should be perspicuous. Historically, the simulation of human decision making has been dominated by sophisticated rule-based approaches. Though such models can engender realistic behavior, they are very complex (often involving thousands of production rules) and difficult to develop. This added complexity is unwelcome in the CGF environment where the simulations are already enormously complex. As our Phase I work has progressed we have realized that we must develop our decision modeling tool as a means to an end and not an end in itself, and this has imposed a healthy sense of parsimony on our model architecture. Finally, a tool that is used to develop CGF simulations for analysis should at least have the potential to provide meaningful data about the decision making process—even if that process is not the object of investigation in every simulation. Once again, even if a rule-based approach engenders realistic behavior, insofar as human decision making is thought *not* to be rule-based, such models will fail to reveal anything about the actual human process they ostensibly represent. Throughout our Phase I efforts we have tried to reflect actual human decision making processes in our model.

4.3 Task 3 – Design an integrating architecture

This task received the bulk of our attention during Phase I. As we indicated above, we now have the basic architectural relations in place to link our model of decision making with a model of uncertainty (via a multiple trace model of LTM) and a model of stress (via a workload model). Although we have been conservative in the variety of effects we have implemented, we have represented an important theoretical connection between the effects of stress and uncertainty in our model. We have also begun to explore an architecture that will allow us to embed our models in HLA-compliant simulations. Finally, with an eye toward a generic and flexible model of RPD, we have consciously limited the amount of context-specific hard-coding in our initial test bed model.

4.4 Task 4 – Identify gaps in modeling architecture

At the outset of our Phase I work, we identified the need for a pattern-recognition routine capable of dealing with uncertain information as the most significant gap in our modeling architecture. As it happened, we found an elegant solution to the problem of missing information in the recognition routine we use to access our LTM. Indeed, as we discussed

above, the real difficulty in dealing with uncertain information was that of modeling a "subjective" response in the synthetic decision maker.

Still, we have uncovered new gaps in our architecture. Most notably, we have yet to implement a robust model of exogenous stressors (e.g., heat, noise, fatigue). This is due more to a lack of time than any outstanding theoretical issues. In fact, implementing the degradation functions discussed in our original proposal will be straightforward; in exactly the same way we assign workload values to each of the cue-updating tasks, we can affect the time it takes to perform the update as well as the accuracy of that update according to a skill degradation multiplier. Thus the information stored in the working memory can be more-or-less timely and more-or-less accurate. In turn, the echo that returns from LTM will reflect a more or less accurate assessment of the situation.

A second, more significant gap is that of "populating" LTM with experience. This is really a combination of three problems. First, we must devise some way of supplying our synthetic decision maker with a store of experiences with his environment. In more concrete terms, we need to produce a collection of traces that can be used to produce an echo during the decision making process. We explored two possibilities during Phase I: the first was to create LTM by hand—something we wanted to avoid on the basis of past experience—and the second was to start with *tabula rasa*, so to speak, and store the results of multiple runs through a simulation as the traces in LTM. In this way we would eventually store enough "experience" to form coherent echoes. The latter alternative holds promise as a natural analogue for "learning from experience," but it also introduces a second problem, namely, how do know when we have added enough traces to capture regularities in the synthetic environment but not so many that the echoes from LTM wash out in noise? Indeed, one of the shortcomings of our multiple trace model of LTM is that, in general, the larger LTM becomes, the more likely it is that several, perhaps unrelated, traces will contribute to the echo. We are currently addressing this issue under our contract with NAWCTSD, and we are confident that whatever solution we find will translate directly to our model of decision making under stressful and uncertain conditions.

Questions about model validation have also surfaced during our Phase I work. Obviously, we will not address these questions by filling gaps in our model architecture, but we must address them nonetheless. In particular, we must decide how we will evaluate process effects in our model as we implement the effects of stress and uncertainty. Up to now, we have often found ourselves relying on face validity to evaluate our implementation. We must also be able to determine when LTM has become "convergent" (i.e., when there are just enough traces to reflect regularities in the environment, but not too many). Should the optional task be awarded, we will explore these questions.

4.5 Develop a functional specification for our model

See appendix.

4.6 Develop a user-interface prototype for our modeling tool

Our most recent Phase I efforts have been devoted to the development of a user-interface that will facilitate the construction of our decision models in HLA-compliant simulation

environments. This interface will be the essential link between our decision-making model and a CGF such as JSAF. The interface will allow the analysts to specify the type of entity making the decision, the type of decision being made and the information necessary to make such a decision. More importantly, the interface will serve as a front end for the middleware that will provide the mapping between the information the analyst specifies and the data that is actually exchanged during the simulation. The middleware will be developed in accordance with HLA specifications and will use the simulation management interactions (SIMAN) of the Real Time Platform Level Reference Federated Object Model (RPR FOM) in order to pass necessary data as specified by the analyst.
A graphical user interface will guide an analyst through a series of forms that collect the information necessary to drive an RPD decision model. We describe these steps in detail below.

RPD Model Prototype

The opening screen (Figure 7) introduces the user to the RPD Model and includes various options in the menu bar including "New RPD Model" and the option to "Open Existing". With these options, a user can start from scratch, completing the necessary fields in each of the successive forms and then save his work so that the same model can be opened again later and modified as necessary without having to begin the whole process anew.

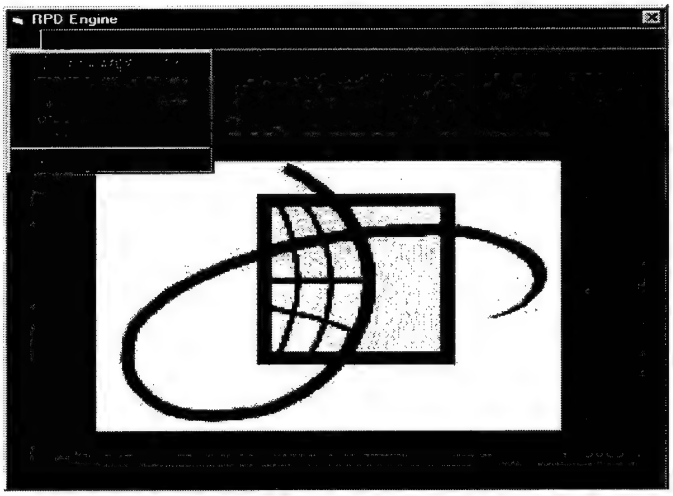

Figure 7 Intro screen for RPD Model Prototype

Step 1: Select Decision Making Entity

After selecting either the "New" or "Open" option from the File menu, the user will be prompted to select the main entity of interest (Figure 8); this will be the decision-making entity. The analyst will select one item from the entities available through JSAF using a checkbox list. (See Figure 8)

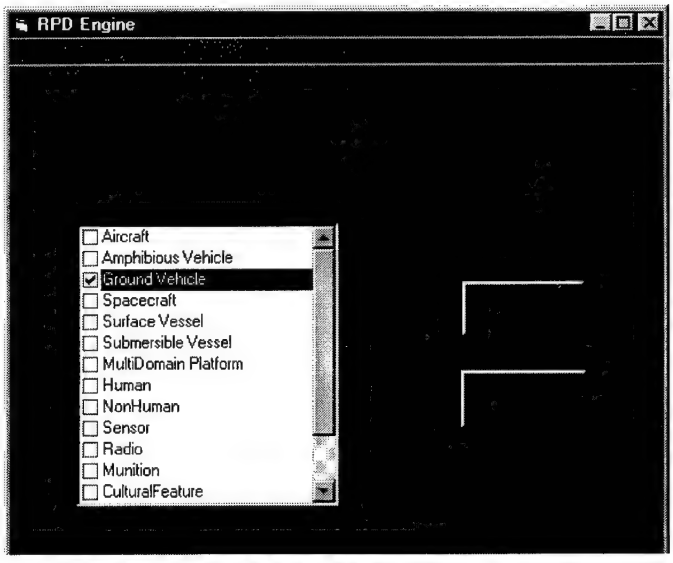

Figure 8 Step 1 – Select Decision Making Entity

Step 2: Select the decision of interest

After selecting the decision-maker, the user will select the decision to be made by the RPD Decision Server. (See Figure 9)

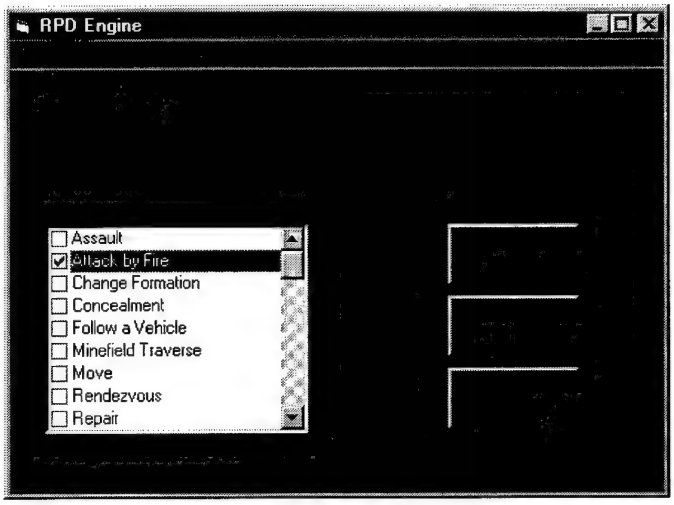

Figure 9 Step 2 – Selecting the Decision of Interest

Step 3: Select entities germane to the decision

Presumably, the decision to be made will depend on the behavior of other entity types in the synthetic environment. Thus, we allow the user to select again from the available entities in JSAF. (See Figure 10).

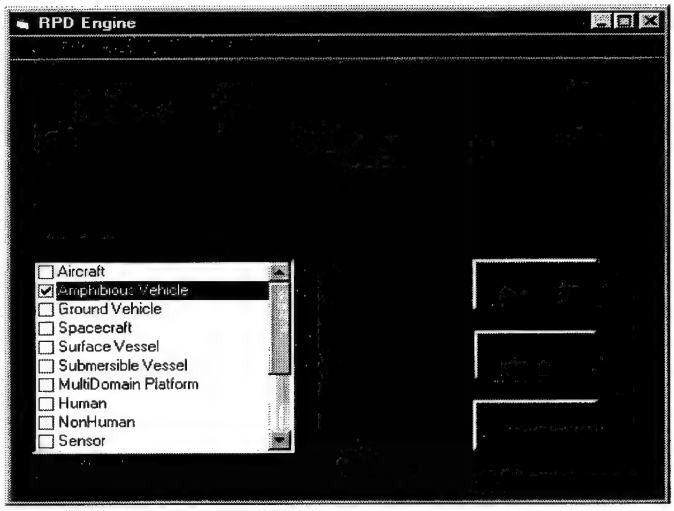

Figure 10 Step 3 – Selecting influencing entities

Step 4: Select input cues for the decision-making entity

The next step for the analyst in the process of creating an RPD model is to designate the attributes of the decision-making entity that cue the decision (Figure 11). These can be thought of as information internal to the entity that must be assessed in order make the decision. Provided in step 4 is a list of all attributes available through the RPR FOM that pertain to the decision making

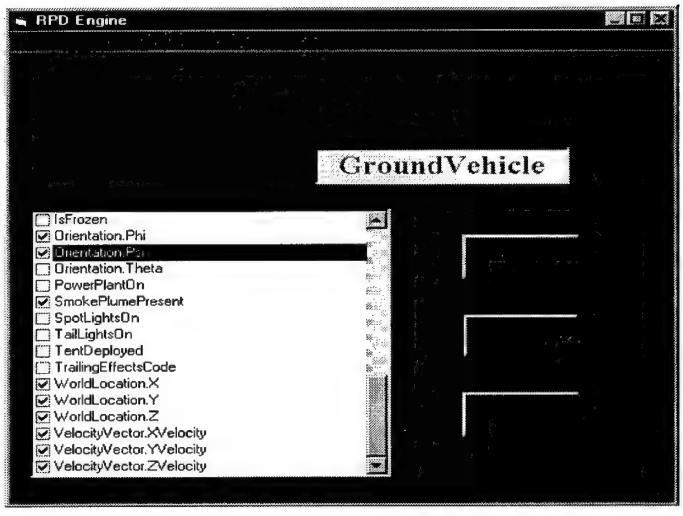

Figure 11 Step 4 – Selecting decision-maker's cues

entity. These include attributes inherent to the entity (i.e. WorldLocation, AccelerationVector) as well as attributes that belong to interaction classes that could be carried out by the entity (Collision.CollisionLocation, WeaponFire.MunitionType).

Step 5: Select input cues for influencing entity

Since the decision making entity must also be aware of what other entities are doing, we prompt the user to select the germane attributes of the entities selected in Step #3. These attributes can be thought of as external information that cues decision making behavior.

Together, the internal and external cues form the set of features used to individuate situations in LTM, and are thus the contents of the agent's "situation awareness." (See Figure 12)

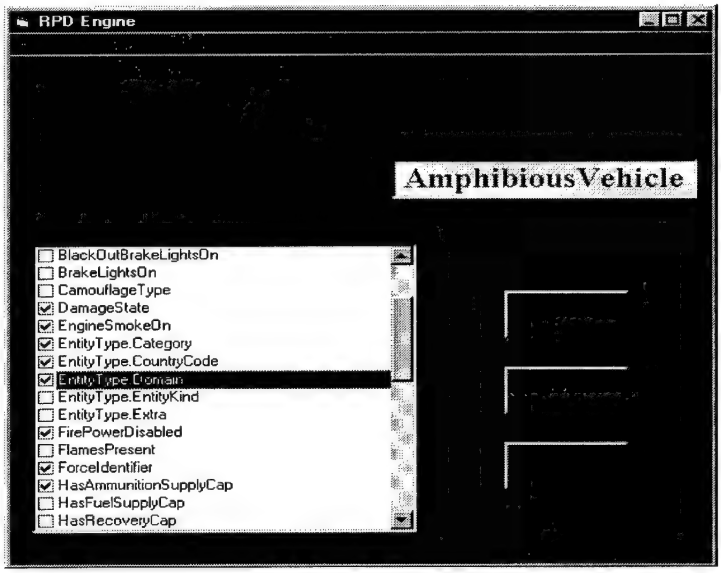

Figure 12 Step 5 – Selecting influencing entities' cues

Steps 6a and 6b: Cue variable specifications

In order to model the effects of stress, we allow the analyst to determine the workload type and value associated with the act of perceiving each cue (both internal and external). In addition, the analyst can specify an "attention management value" and decay rate (as discussed above). The attention management rank determines the relative likelihood of a particular cue being sampled (relative with respect to the other cues being sampled). In fact, the analyst specifies two such values for each cue: one value to be used during low-stress (i.e., non overloaded) conditions, and another value to be used during high-stress (i.e., overload) conditions. In this way we can represent the effects of attention narrowing discussed above. Likewise, during low-stress conditions, the decay rate

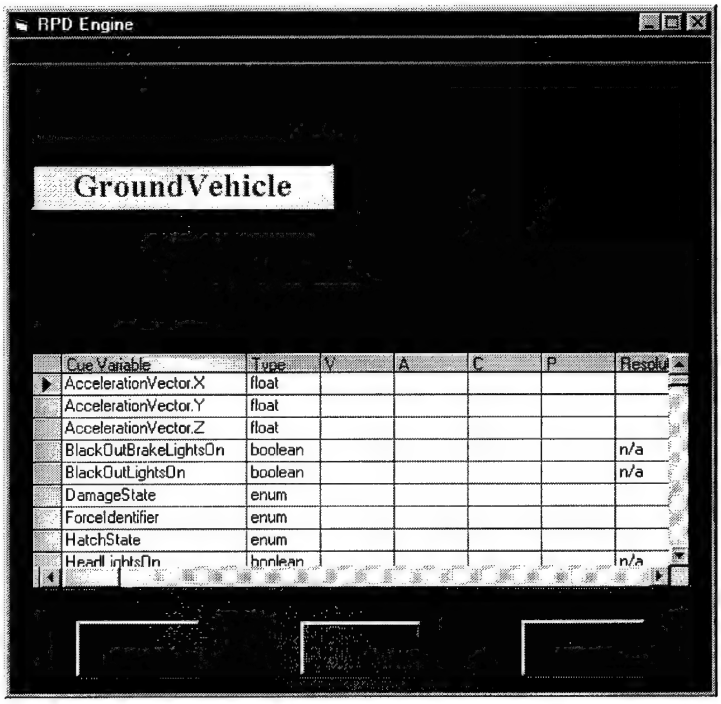

Figure 13 Step 6a – Database grid for cue variable specifications such as workload, attention management, and decay rate

specified by the analyst will ensure a forgetful decision maker, while in high-stress conditions, the incremented decay rates will capture the effects of decreased working memory capacity. (See Figure 13)

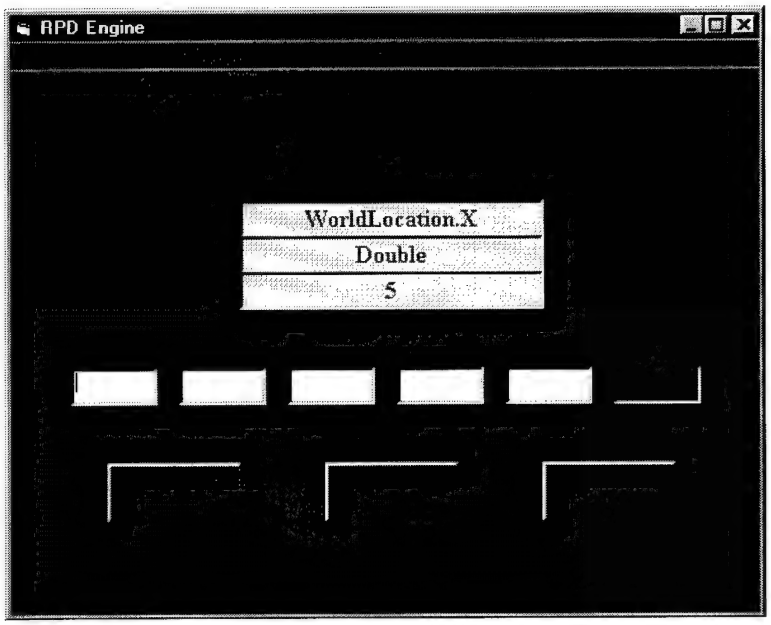

Figure 14 Step 6b – Boundary specification for variable encoding

At this point, we expose the analyst to an obnoxious implementation detail. Due to the mechanics of our LTM model, the synthetic decision maker is limited to a three-bit memory, so to speak. Each bit in LTM encodes a feature that is either asserted (value of 1), denied (value of –1) or unknown (value of 0). Consequently, any non-Boolean attribute value (i.e., integer or float) must be represented in a form the synthetic decision maker can understand in these three-bit terms. Our proposed approach here is to represent non-Boolean attribute values in terms of discrete scales. For example, an attribute that can take on any value between say, 0 and 100, will be represented in LTM by, say, a low-medium-high scale where values less than 33 are encoded as low, values between 33 and 66 are encoded as medium, and values greater than 66 are encoded as high. In this way we are able to maintain the three-bit memory structure, and yet still represent non-Boolean attribute values.[4] Unfortunately, this also means the analyst must decide how fine a scale to use to represent each non-Boolean attribute value, and, likewise, where to impose cut-off points on the scale. The GUI shown in (Figure14) is intended to facilitate this process: the analysts specifies the "Resolution" of the scale and the "Boundary" values that will set cut-off points between intervals on the scale.

Having stepped the user through this process, the middleware will impose a trace structure on LTM and ensure that the RPD decision server receives the necessary information from the various federates in the CGF simulation.

5. Preliminary Results and Outstanding Problems

With a working test bed model in place, we have begun model analysis in tandem with continuing development. The main issues we hope to answer are whether we can add traces to LTM on the fly, so to speak, and have our synthetic decision maker "learn" the

[4] Note, there are two reasons this problem is not solved simply by adopting a base-3 numeral system. First, it is simplifies the encoding if each trace in LTM has the same "width" (i.e., each trace must use the same number of bits to represent situations). Second, past a certain point, it is advantageous to limit the overall width of traces in LTM.

appropriate associations between situations and COAs, and whether this approach will guide similar situations to converge on appropriate courses of action. Put in slightly more abstract terms, we need to determine when LTM is *converging* in the sense that a given situation will eventually induce a composite echo that represents the recognition of an appropriate COA for that situation. The analysis we have performed and will continue to perform is guided by several questions: Are similar traces grouped uniquely according to COA? Is that unique COA appropriate? How long does it take the decision maker to learn associations or similarly for the model to "spin-up"? Are there visible effects of stress and uncertainty on the decision maker's performance such as a decline in appropriate situation-COA associations?

There are five possible courses of action for our synthetic driver: GO, STOP, SPEED-UP, SLOW-DOWN, and MAINTAIN. The driver starts off with no experience (i.e., there are no traces in LTM) and, at first, must guess which of these COAs is appropriate for the situation at hand. After the driver implements a COA, a *success value* is assigned to that situation-COA experience according to the objective outcome of the COA. Note that an optimal decision is assigned a success value of '1' while an abysmal choice of COA is assigned a success of '-1'. For example, stopping when the driver is very far from the light is bad (hence receiving a large negative success value), while driving through a green light is good (hence receiving a large positive success value). At the end of each episode, the situation-COA pair is stored together with the success value as a new trace in LTM. As the driver encounters subsequent situations, both by-products and success values are recalled from LTM; at this point, rather than choose randomly from the available COAs, the driver can use the successes and failures of his past experiences to make his choice.

In this context, we can address questions about the convergence of LTM by analyzing the changing success values assigned to each episode as the driver gains experience. We expect that success values will improve with time only if similar situations are eventually grouped uniquely with an appropriate (i.e., successful) COA.

After stepping through several model runs, we found that many of the echoes returned were very flat – rarely did an echo signal recognition toward a unique COA. For the most part, all echo content values were between the two noise thresholds, indicating that either LTM was too noisy or that our calculation of the noise thresholds was flawed. We looked carefully at our approach for computing the noise thresholds and did in fact decide to alter our method. Initially, we were calculating the noise by figuring the echo content values from a completely random LTM, counting *every* trace in LTM as a trial. The problem with this is that some traces are orthogonal to the probe and thus do not contribute to the echo at all. Eventually, we decided not to count these non-contributing traces when we compute noise thresholds. In particular, we decided not to count non-contributing traces toward the number of trials that fix the binomial distribution we use to model noisy echoes. We also decided that the average success should be computed only over the contributing traces (as opposed to averaging success over all the traces in LTM). These changes, collectively, contributed to lower, more realistic noise thresholds.

With lower noise thresholds, it is evident that situations are in fact converging on courses of action. More often than not, a clear, unique course of action is recognized as its echo content value is significantly positive – exceeding the positive noise threshold. Once again stepping through model runs one at a time, we investigated the echo and the relationship of its content values to the positive and negative thresholds. Amazingly, with no rules, or previous experiences to feed LTM initially, the model behaved very well. With or without stress and uncertainty, the driver is very good at distinguishing between intermediate and final courses of action. He knows when to stop speeding up, or slowing down, and when to go or stop.

In order to assess both the overall performance of the decision maker as well as the effects of stress and uncertainty on model output, we ran ten batches of 500 runs for a driver influenced by stress and uncertainty and ten batches for a driver not influenced by stress and uncertainty. In the discussion following, a batch refers to a 500-run simulation, and a segment refers to a 100-run block of runs within a batch. The results for the two different sets of simulations are summarized in Appendix B.

In both sets of results, mean success values for all but two segments of 100 runs are positive. This indicates to us that the driver is in fact learning and similar situations are converging on optimal or sub-optimal courses of action. This is further evidenced by the comparison of the number of decisions with positive success values to the total number of decisions for each 100 runs. Out of the 100 segments in the 20 batches combined (the 10 *with* stress and uncertainty and the 10 *without*), only 6 segments yielded more negative success values than positive ones. Therefore, 94 out of 100 sets of 100 runs were predominantly successful. This suggests that our decision maker is learning and is forming appropriate associations between situations and courses of action (significance test are forthcoming).

Looking at the trends in the success values from the first 100 runs to the fifth 100 runs inside each batch (each simulation of 500 successive runs), we do not notice any consistent indication that success values are continuously increasing with experience. However, the first segment of 100 runs in almost every batch exhibits the lowest mean success compared to the other four segments of 100 runs. After the first segment, mean success values roughly increase until the second or third segment at which point they seem to stabilize or slightly taper off. We interpret this as an indication that our spin-up phase occurs in the first 100 to 200 runs and soon thereafter the model reaches its optimal performance.

To determine whether any improvements between the first segment of 100 runs and later segments of 100 runs inside a given batch were statistically significant, we employed a sign test. We used the sign test to compare mean success values between the following pairs of segments in each batch:

1^{st} and 2^{nd} 100 runs
1^{st} and 3^{rd} 100 runs
1^{st} and 4^{th} 100 runs
1^{st} and 5^{th} 100 runs

In the model simulations *with* stress and uncertainty, 18 out of 40 comparisons indicated a significant improvement from the first segment of 100 runs (the last column in Table 1 and Table 2 in Appendix B indicates whether or not a given segment had a significantly higher mean success value than the first segment in the same batch). As for the model runs *without* stress and uncertainty, only 13 out of the 40 comparisons depicted a significant improvement. Although this might make it seem as though there is no consistent improvement with experience, there are many factors to take into account. The sign test was used only to detect if an improvement was *significant.* All of the negative results from the sign test do not necessarily indicate declines in performance; they only tell us that the mean success of the latter segment in the given batch was not *significantly* greater than the mean success of the first segment of 100 runs. Another factor to take into account is that the sign test does not reflect segments' mean success values, but rather the relationship between success values. The fact that 98% of the segments tested had positive mean success values suggests that the decision maker is learning. Lastly, we arbitrarily chose the segment size of 100 runs. We assumed that the first 100 runs would have poor performance due to the model's spin-up phase and therefore would be a good segment against which to measure the performance of later segments in a batch. However, we often found that the first segment was quite successful, therefore leaving little room for improvement in later segments. Consequently, the low proportion of significant improvements revealed by the sign test comparisons could be due to the fact that there was not a whole lot of room for improvement.

We have discussed overall model performance in a general sense, but we also need to differentiate between the results for the different *types* of model simulations. As noted earlier, we ran 10 batches of 500 runs *with* the routines for stress and uncertainty and we ran 10 batches *without* the stress and uncertainty routines. Surprisingly, the overall average success value for the simulations *with* stress and uncertainty was 0.381 whereas the average success for simulations *without* stress and uncertainty was lower at 0.275. Looking solely at the proportion of decisions with positive success values, again the simulations using stress and uncertainty seemed to do somewhat better with 68% of the decisions showing positive success values while simulations without the effects of stress and uncertainty yielded successful decisions 62% of the time.

These results may seem awkward; one would expect that the effects of stress and uncertainty actually hinder the performance of a decision maker yet our preliminary results indicate otherwise. This could primarily be due to the fact that in our first attempt at modeling stress and uncertainty we have assigned workload values, decay rate, and overload threshold values arbitrarily in order to get the model running with the complete routines for stress and uncertainty. Because we do not yet have good data from subject matter experts to feed to the workload and memory decay portions of the model, we would expect the results of stress and uncertainty to be inconsistent, unpredictable, and perhaps even contradictory.

6. Summary

The goal of our Phase I research was to explore the possibility of improving behavioral realism in CGFs by improving the representation of human decision making in stressful and uncertain conditions. Our approach was premised on the assumption that an improved representation of decision making would begin with the most current theories of human decision making. Consequently, we found ourselves reviewing the literature on Naturalistic Decision Making, and the model of Recognition Primed Decision making in particular. In this respect, our work has diverged sharply from traditional, rule-based approaches. Rather than implementing utility functions or multi-attribute analyses, our Phase I work depended on models of situation awareness, long term memory and recognition. Moreover, we had to link these models in such a way that they could accommodate additional models of stress and uncertainty. Integrating these component models in a principled way has been one of the largest challenges of our Phase I work. The difficulties here are compounded by the fact that the theory concerning the effects of stress and uncertainty on the decision making process is fractured. But despite this fact, we believe we have developed an architecture that will capture the effects of uncertainty in a natural way, reflect changes in endogenous stress (via a workload measure) and extend easily to capture the effects of exogenous stressors.

Because we have tied our efforts so closely to a cutting-edge theory of what human decision makers actually do, we see potential in our approach that has been wanting in more traditional approaches. For instance, we believe that the structure of our models is far more perspicuous than that of most rule-based models of decision making. This has the potential to streamline model development and, at the same time, should give the analyst a clearer picture of what's going on during model execution. Moreover, unlike a rule-based system that might depend on literally thousands of production rules, the RPD mechanisms we have modeled here are relatively simple and this, together with the performance server architecture we've proposed, will make the integration of our decision models into distributed simulations more straightforward—a welcome relief in the context of CGFs where no one needs to see additional complexity. Finally, because we have represented the decision making process that humans are actually thought to use, our model has the potential to provide information to the analyst about the decision making process that would otherwise be unavailable through simulation.

We have laid a solid foundation for Phase II work in the last six months. We are now ready to model additional exogenous stressors, and to begin analyzing the overall impact of stress and uncertainty in our model. With these results in hand, we would be ready to move beyond our test bed model and begin work on modeling decisions within a CGF environment. We expect this work will lead to additional refinements in our model architecture, but more importantly, it will provide the context in which we will develop our middle-ware. With the middleware and the attendant interfaces in place, we will have created a stand-alone tool to model human decision making in stressful and uncertain conditions. This technology will reduce the need for human-in-the-loop simulations and thus provide the analyst with a powerful source of data.

References

Appley, M. and R. Trumbell (1976). Psychological Stress. New York, Appleton-Century-Crofts.

Bierbaum, C. R., S. M. Szabo, et al. (1987). A comprehensive task analysis of the UH-60 mission with crew workload estimates and preliminary decision rules for developing a UH-60 workload prediction model. Fort Rucker, AL, Anacapa Sciences, Inc.

Cannon-Bowers, J. A., E. Salas, et al. (1996). "Establishing the Boundaries of a Paradigm for Decision-Making Research." Human Factors **38**: 193-205.

Cox, T. (1975). Behavioural pharmacology. Psychology Today. W. E. R. Gillham. London, English University Press.

Gillis, P. D. and S. R. Hursh (1999). Using Behavior Moderators to Influence CGF Command Entity Effectiveness and Performance. 8th CGF-BR Conference.

Hammond, K. R. (2000). Judgments Under Stress. New York, Oxford University Press.

Hintzman, D. L. (1986). ""Schema Abstraction" in a Multiple-Trace Memory Model." Psychological Review **93**(4): 411-428.

Keinan, G. (1987). "Decision Making Under Stress: Scanning of Alternatives Under Controllable and Uncontrollable Threats." Journal of Personality and Social Psychology **52**(3): 639-644.

Klein, G. (1996). The Effects of Acute Stressors on Decision Making. Stress and Human Performance. J. E. Driskell and E. Salas. Mahwah, NJ, Lawrence Erlbaum Associates Inc.

Klein, G., J. Schmitt, et al. (1996). Fighting in the Fog: A Study of Uncertainty in the U.S. Marine Corps. Dayton, Klein Associates.

Lacey, J. I. (1967). Somatic response patterning and stress: Some revisions of activation theory. Psychological Stress. M. Appley and R. Trumbell. New York, Appleton-Century-Crofts: 14-42.

Lazarus, R. (1966). Psychological Stress and the Coping Process, McGraw-Hill.

Levine, S. and N. A. Scotch (1970). Social Stress. Chicago, Aldine Publishing.

McCracken, J. H. and T. B. Aldrich (1984). Analyses of selected LHX mission functions: Implications for operator workload and system automation goals. Fort Rucker, AL, Army Research Institute Aviation Research and Development Activity.

McGrath, J. E. (1970). Social and Psychological Factors in Stress. New York, Holt, Rinehart and Winston.

Noble, D. (1993). A Model to Support Development of Situation Assessment Aids. Decision Making in Action: Models and Methods. G. Klein, J. Orasanu, R. Calderwood and C. E. Zsambok. Norwood, New Jersey, Ablex Publishing: 287-305.

Schmitt, J. F. and G. A. Klein (1996). "Fighting in the Fog: Dealing with Battlefield Uncertainty." Marine Corps Gazette: 62-69.

Tversky, A. and D. Kahneman (1974). "Judgment under uncertainty: Heuristics and biases." Science **185**: 1124-1131.

Yerkes, R. M. and J. D. Dodson (1908). "The relation of strengtrh to rapidity of habit formation." Journal of Comparative and Neurological Psychology **18**: 459-482.

Appendices

A. Functional Specifications and Test Bed Network Diagram

The test bed model consists of two main components: the first is the task network that models the driving environment (both in terms of the objective features of the environment and the tasks the driver performs while interacting with the environment) and the second is an object-oriented model of LTM. The task network is a stoplight model that was created under an independent contract for NAWCTSD.

Task Network

When the model begins executing, the driver samples from the environmental features (i.e., the driver notices cues in the environment) in the "driving tasks" section of the task network (tasks 13, 14, 15, 20, 52, 59, 73 in Figure 15). At the same time, the task network imposes distractions that compete for the driver's attention. Cues are sampled inside the "driving tasks" according to a set of probabilities that dictate the relative likelihood of performing each update task. In this way, we can affect the driver's "attention management" strategy. Visual, auditory, cognitive, and psychomotor (VACP) component workload values are incremented inside each task accordingly and the total workload is repeatedly updated as the sum of the four component's workload values. In order to represent the effects of inherent time pressure on cognitive workload, we implemented a linear function to increment cognitive workload as a function of the time left in the decision scenario. If, in fact, the driver is overloaded due to stress and/or uncertainty, a new set of cue sampling probabilities that reflect the driver's narrowing of attention is enforced. After each update task is performed, and assuming that the update was not precluded by a distraction, the sampled cue value is stored along with a time stamp in a "working memory" array. These values form the basis for the driver's "situation awareness". A continuously repeating function compares the time stamps for each of the cue values with the current time and if the difference between the two exceeds the given cue's decay time (a predetermined value) then that cue is essentially forgotten (i.e., the cue value goes to zero in the working memory array).

The driver continues to loop through the driving tasks and update "working memory" sub-network (network #21) until he recognizes that he is at a critical point and needs to start making decisions. This sends the driver into the RPD sub-network (network #32). Inside the RPD subnetwork, the contents of the working memory array are passed to the LTM module (discussed in more detail below), at which point execution in the task-network is paused and a variety of methods are called to produce a Hintzman-like echo from LTM (these methods are discussed below). This echo represents the *by-products of recognition* (expectancies and COA in particular). If none of the returned COAs exceed the noise threshold then an 'uncertainty' flag is set which causes cognitive workload to increase. Similarly, if more than one COA is recognized (i.e., more than on COA exceeds the noise threshold), then an 'ambiguous' flag is set and cognitive workload increases. The by-products are passed back to the task network and execution resumes with the driver comparing the evolving state of his environment to the expectancies just retrieved from LTM.

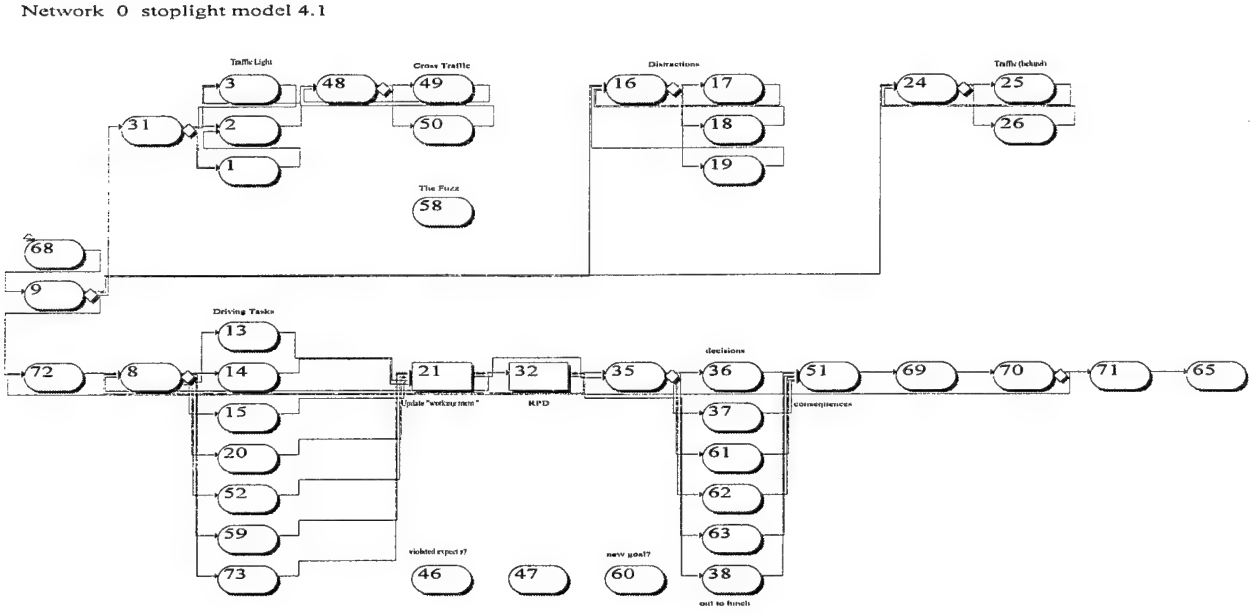

Figure 15 Micro Saint task network diagram for the traffic model

If the expectancies are satisfied, then the driver implements the recognized COA (performing one task among tasks 36, 37, 61, 62, 63, and 38). If, however, the expectancies are violated, the driver will re-assess the situation; that is, the driver will loop through a new cycle of cue updates and working memory updates, probe LTM a second time and then evaluate the most recently returned set of expectancies. This process continues until a set of expectancies is satisfied or the driver simply runs out of time. In either event, the driver implements the most recently recognized COA.

Once the COA is implemented, the driver's decision is evaluated against the current state of the environment (task #51). Success values range from very unsuccessful to very successful depending on the consequences of the driver's decision. For example, if the driver has decided to run the light and there happens to be traffic in the intersection, then the driver's decision is deemed very unsuccessful. By contrast, if the driver decides to stop at a red light, then his decision is evaluated as moderately successful. This success value is passed back to LTM with the contents of working memory that cued the COA that was, in fact, implemented. These values, together with a set of expectancy values representing the state of the environment at the time the decision was implemented, are stored as a new trace in LTM, and thus add to the driver's experience. At this point execution halts.

An Object Oriented Model of LTM

Our implementation of Hintzman's multiple trace model of LTM depends on four objects, a "learning" algorithm and an interface. In particular, we have a long term

memory object, a current situation object, an echo object, and an object that represents the current state of the environment. We briefly describe each object and their methods below.

Current Situation Object

The current situation object stores the values from the task network representing the agent's working memory (WM). Once the agent has a complete representation of the current situation in WM, a method is invoked to probe LTM (a two dimensional array stored in the long term memory object discussed below). At this time similarity and activation values are computed (with respect to the current situation) for each trace in LTM.

Long Term Memory Object

Long term memory is a two dimensional array variable belonging to this object. Each row in the array represents a different situation and the associated by-products of recognition. The LTM object also supports methods for populating itself (using either "rules" specified by the analyst, past experience or by adding traces on the fly during model execution).

Echo Object

Using the similarity and activation values calculated within the current situation object, the echo object computes the echo intensity and the echo content. The echo content represents the byproducts of recognition (expectancies and course of action). This object also contains a method for determining noise thresholds. These thresholds are called upon later in the learning algorithm before the byproducts are returned to the task network.

Learning Algorithm

The learning algorithm determines the COA and expectancy values that are returned to the task network by comparing echo content values with their respective noise thresholds (different thresholds are calculated for COAs and for expectancies). For the COA the most successful significant echo is returned. If none of the COAs are significantly positive, and one or more are significantly unsuccessful, then the algorithm chooses randomly among the remaining echoes that are neither significantly unsuccessful nor significantly successful. To determine the expectancy values from the expectancies' echo content values, a similar comparison is performed between expectancy thresholds and expectancy echo content values. If a content value is greater than the positive threshold, then that expectancy value is returned to the task network as a '1' (indicating that the expectancy has been asserted positively—i.e., expect this). If, on the other hand, the content value is less than the negative threshold, then it is returned as a '-1' (indicating that the expectancy has been negatively asserted—i.e., expect this not, or rather, this shouldn't happen) and if the content value is between the two thresholds, the expectancy value is returned as a '0' (indicating that nothing is to be expected of this feature).

World State Object

This object stores values from the task network representing the state of the environment a few seconds after the expectancies have been recognized. These values are stored in LTM as will eventually influence the expectancies the agent form in later situations. In

addition to the expectancies, this object also stores the success value computed at the end of a run.

Software Interfacing Object

Communication between the task network and the LTM module is accomplished through Microsoft's Component Object Model (COM). COM is an open architecture designed to accomplish cross-platform communication. Using supporting libraries and conventions, COM allows the interaction between two different pieces of software in a consistently object oriented manner. The software interfacing object listens to COM for the variables being sent from the task network. With its various *receive-variable* procedures, all of the necessary variables coming from COM via the task network are recognized and assigned to the corresponding Visual Basic variables.

B. Preliminary Results Data

Simulation	Segment of Runs	Mean Success	Total Decisions	# Decisions Resulting in Positive Success	Sign Test r	Sign Test: Significant Improvement?
1	1st 100	0.154	280	156	---	---
	2nd 100	0.166	261	149	0.51	No
	3rd 100	0.318	251	163	0.60	Yes
	4th 100	0.244	260	157	0.55	No
	5th 100	0.157	321	182	0.51	No
2	1st 100	0.094	284	157	---	---
	2nd 100	0.308	284	185	0.61	Yes
	3rd 100	0.452	245	177	0.68	Yes
	4th 100	0.463	269	195	0.69	Yes
	5th 100	0.455	253	183	0.70	Yes
3	1st 100	0.511	303	227	---	---
	2nd 100	0.455	302	219	0.45	No
	3rd 100	0.546	316	243	0.53	No
	4th 100	0.538	328	251	0.55	No
	5th 100	0.557	277	214	0.54	No
4	1st 100	0.230	342	207	---	---
	2nd 100	0.524	329	250	0.67	Yes
	3rd 100	0.498	326	244	0.66	Yes
	4th 100	0.433	310	221	0.61	Yes
	5th 100	0.498	321	239	0.65	Yes
5	1st 100	0.425	423	298	---	---
	2nd 100	0.359	462	312	0.47	No
	3rd 100	0.346	444	297	0.44	No
	4th 100	0.327	462	303	0.43	No
	5th 100	0.370	423	287	0.47	No
6	1st 100	0.118	189	103	---	---
	2nd 100	0.311	167	108	0.60	Yes
	3rd 100	0.371	171	116	0.63	Yes
	4th 100	0.314	199	129	0.58	No
	5th 100	0.423	213	151	0.64	Yes
7	1st 100	0.402	322	223	---	---
	2nd 100	0.480	310	229	0.56	No
	3rd 100	0.484	298	220	0.56	No
	4th 100	0.439	297	212	0.56	No
	5th 100	0.519	295	223	0.57	Yes
8	1st 100	0.404	276	192	---	---
	2nd 100	0.393	291	201	0.50	No
	3rd 100	0.399	227	158	0.51	No
	4th 100	0.304	275	176	0.45	No
	5th 100	0.414	210	147	0.51	No
9	1st 100	0.294	428	274	---	---
	2nd 100	0.402	403	280	0.57	Yes
	3rd 100	0.400	469	325	0.56	No
	4th 100	0.344	448	297	0.56	No
	5th 100	0.359	434	291	0.56	No

Simulation	Segment of Runs	Mean Success	Total Decisions	# Decisions Resulting in Positive Success	Sign Test r	Sign Test: Significant Improvement?
10	1st 100	0.258	227	143	---	---
	2nd 100	0.421	212	149	0.61	Yes
	3rd 100	0.481	192	141	0.66	Yes
	4th 100	0.426	189	134	0.63	Yes
	5th 100	0.450	210	152	0.63	Yes

Table 1 Results for 10 batches of 500 model runs each. In all of the 10 batches, stress and uncertainty *were* included.

Simulation	Segment of Runs	Mean Success	Total Decisions	# Decisions Resulting in Positive Success	Sign Test R	Sign Test: Significant Improvement?
1	1st 100	0.002	274	132	---	---
	2nd 100	0.108	280	153	0.52	No
	3rd 100	0.090	327	167	0.56	No
	4th 100	0.121	291	150	0.55	No
	5th 100	0.158	290	158	0.58	Yes
2	1st 100	0.074	473	230	---	---
	2nd 100	0.126	518	266	0.54	No
	3rd 100	0.160	423	229	0.54	No
	4th 100	0.225	448	249	0.58	Yes
	5th 100	0.162	394	214	0.56	Yes
3	1st 100	0.295	283	181	---	---
	2nd 100	0.312	253	163	0.55	No
	3rd 100	0.441	219	156	0.61	Yes
	4th 100	0.273	266	167	0.50	No
	5th 100	0.330	263	174	0.54	No
4	1st 100	0.263	240	149	---	---
	2nd 100	0.447	218	157	0.60	Yes
	3rd 100	0.425	207	147	0.61	Yes
	4th 100	0.410	223	157	0.58	No
	5th 100	0.373	203	138	0.57	No
5	1st 100	0.245	275	168	---	---
	2nd 100	0.378	229	156	0.55	No
	3rd 100	0.328	308	202	0.55	No
	4th 100	0.377	249	169	0.55	No
	5th 100	0.293	256	163	0.53	No
6	1st 100	0.295	255	164	---	---
	2nd 100	0.380	247	172	0.52	No
	3rd 100	0.389	231	155	0.57	Yes
	4th 100	0.339	231	150	0.52	No
	5th 100	0.546	195	149	0.62	Yes
7	1st 100	0.026	287	136	---	---
	2nd 100	0.311	275	171	0.63	Yes
	3rd 100	0.046	241	114	0.48	No
	4th 100	0.300	249	151	0.62	Yes
	5th 100	0.385	245	163	0.66	Yes

Simulation	Segment of Runs	Mean Success	Total Decisions	# Decisions Resulting in Positive Success	Sign Test R	Sign Test: Significant Improvement?
8	1st 100	0.362	274	191	---	---
	2nd 100	0.381	297	210	0.51	No
	3rd 100	0.312	246	168	0.47	No
	4th 100	0.361	278	190	0.54	No
	5th 100	0.471	237	173	0.59	Yes
9	1st 100	0.383	232	157	---	---
	2nd 100	0.397	223	154	0.52	No
	3rd 100	0.500	248	183	0.56	No
	4th 100	0.287	236	150	0.47	No
	5th 100	0.350	247	165	0.48	No
10	1st 100	0.074	280	147	---	---
	2nd 100	0.263	259	154	0.58	Yes
	3rd 100	-0.050	253	109	0.46	No
	4th 100	0.222	271	162	0.56	No
	5th 100	-0.015	261	116	0.50	No

Table 2 Results for 10 batches of 500 model runs each. In all of the 10 batches, stress and uncertainty *were not* included.